TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SAMUEL HYYPPÄ

KAIVOSKONEEN AUTONOMINEN NAVIGOINTI GENEERISESSÄ YMPÄRISTÖSSÄ HYÖDYNTÄEN PISTEPILVIÄ

Diplomityö

Tarkastajat: professori Robert Piché,
professori Reza Ghabcheloo
Tarkastajat ja aihe hyväksytty
31. toukokuuta 2017

TIIVISTELMÄ

SAMUEL HYYPPÄ: Kaivoskoneen autonominen navigointi geneerisessä ympäristössä hyödyntäen pistepilviä
Tampereen teknillinen yliopisto
Diplomityö, 81 sivua
Heinäkuu 2018
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Prosessien hallinta
Tarkastajat: professori Robert Piché, professori Reza Ghabcheloo

Avainsanat: kaivoskone, runko-ohjattava, autonominen navigointi, pistepilvet, SLAM, 3D-lasertutka, liikesuunnittelu, mallinnus, simulointi

Nykyinen autonomisten runko-ohjattavien kaivoskoneiden navigointijärjestelmä perustuu 2D-lasertutkiin sekä ympäristön ja reitin opetukseen. Uuden reitin käyttöönottoon kuluvaa aikaa halutaan pienentää. Tämä voidaan tehdä esimerkiksi automatisoimalla liikesuunnittelu, jolloin ei tarvita reitin opetusta ollenkaan. Nykyinen järjestelmä ei kykene automaattiseen liikesuunnitteluun seuraavien heikkouksien takia. Ensinnäkin 2D-lasertutkilla ei voida havaita mittaustason ala- tai yläpuolella olevia mahdollisia esteitä. Toiseksi samaa opetettua ympäristömallia ei voida hyödyntää mahdollisesti muissa ajoneuvoissa. Kolmanneksi tarkka paikannus ei aina onnistu. Neljänneksi järjestelmää ei voida käyttää dynaamisissa ympäristöissä. Automaattista liikesuunnittelua tai reitin mukauttamista ei voida suorittaa riskittömästi käyttämällä vain kahta 2D-lasertutkaa, minkä takia tarvitaan ainakin yksi 3D-lasertutka.

Tulevaisuuden kaivoskoneen navigointijärjestelmän perustana voitaisiin käyttää Krüsin et. al (2017) artikkelissa esitettyjä menetelmiä. Siinä hyödynnetään 3D-pistepilviä liikesuunnitteluun ja maaston arviointiin maassa liikkuville roboteille. Tässä työssä on replikoitu ja muokattu kaivoskoneille sopiviksi osia kyseisessä artikkelissa esitetyistä algoritmeista. Tässä työssä käytetty tunnelin 3D-pistepilvimalli on luotu avoimen lähdekoodin SLAM-algoritmilla (Simultaneous Localization and Mapping) nimeltä BLAM (Berkeley Localization and Mapping). Lisäksi tässä työssä on käytetty kaivoskoneen mallintamiseen ja simuloimiseen WMRDE:tä (Wheeled Mobile Robot Dynamics Engine). Simulaattori on tarkka, modulaarinen ja nopea. Tämän merkitys voi korostua vasta jatkotutkimuksissa, jos käy ilmi, että tässä työssä käytetyt alkeisliikeradat eivät ole riittävän tarkkoja, tai jos halutaan muodostaa turvakerros navigointijärjestelmälle.

Työssä havaitaan, että ICP:hen (Iterative Closest Point) perustuvalla SLAM-algoritmilla on mahdollista luoda paikallisesti tarpeeksi tarkkoja 3D-pistepilviympäristömalleja, joilla voidaan riittävän luotettavasti navigoida tunnelistossa. Kehittämällä työssä käytettyä SLAM-algoritmia Krüsin et. al artikkelin mukaisesti voidaan parantaa myös sen suorituskykyä vielä huomattavasti. Käyttämällä artikkelin menetelmiä voidaan kaivoskoneille toteuttaa autonomisen navigointijärjestelmän vaatima automaattinen liikesuunnittelu.

ABSTRACT

SAMUEL HYYPPÄ: Autonomous Navigation of Underground Mining Vehicle in Generic Environments by Utilizing Point Clouds
Tampere University of Technology
Master of Science Thesis, 81 pages
July 2018
Master's Degree Programme in Automation Technology
Major: Process Automation
Examiners: Professor Robert Piché, Professor Reza Ghabcheloo

Keywords: underground mining vehicle, articulated center pivot steering, autonomous navigation, point clouds, SLAM, 3D lidar, motion planning, modelling, simulation

The navigation system of the current autonomous articulated underground mining vehicles is based on 2D lidars and the teaching of the environment and route. The introduction of the new route is desired to take less time. This could be achieved by automating the motion planning whereupon the teaching is not needed any more. The current system is unable to do automatic motion planning due to the following weaknesses. Firstly, 2D lidars are not able to perceive anything under or above the measurement level. Secondly, the same earlier taught environment model could not be utilized in other vehicles. Thirdly, the accurate localization is not always possible. Fourthly, the system could not be used in dynamic environments. The automated motion planning cannot be done safely by using only two 2D lidars; therefore, at least one 3D lidar is required.

In the future, the navigation system of the underground mining vehicle could be based on the methods presented in the article by Krüsi et al (2017). In that article, 3D point clouds are utilized in motion planning and terrain assessment for ground robots. In this thesis, some algorithms of that article are replicated and edited to be compliant for the underground mining vehicles. The 3D point cloud of the mine tunnels used in this thesis is created by an open source SLAM algorithm (Simultaneous Localization and Mapping) called BLAM. In addition, WMRDE (Wheeled Mobile Robot Dynamics Engine) is used for the modelling and the simulation of the underground mining vehicle in this thesis. The simulator is accurate, modular and fast. However, the meaningfulness of this simulator could not be emphasized until the future work, if it is revealed that the utilized elementary trajectories are not enough accurate or the safety layer is needed for the navigation system.

In this thesis, it is realized that it is possible to create with the SLAM algorithm based on the ICP (Iterative Closest Point) locally accurate 3D point cloud environment models that have sufficient reliability for the navigation inside the mine tunnels. By developing the SLAM algorithm that is used in this thesis according to the article by Krüsi et. al further, its performance can be improved. By using the methods of the article, the automatic motion planning that is required by the autonomous navigation system of the underground mining vehicle can be developed.

ALKUSANAT

Tämä työ on saanut alkunsa vuoden 2017 Yrityspäivillä tapahtuneesta keskustelusta Sandvikin messuosastolla ja sen jälkeisestä yhteydenotosta sekä haastattelusta, jossa kävi ilmi diplomityön aihe suurpiirteisesti. Aihe liittyi kaivoskoneiden automaattiseen reitinsuunnitteluun ja itseoppivaan navigointiin. Tässä työssä ei keskitytä vain 2D-reitinsuunnittelualgoritmeihin kuin aluksi ajattelin, mutta ei kuitenkaan saavuteta itseoppivuutta navigoinnin osalta.

Kesä 2017 kului pääasiassa tutkiessa erilaisia vaihtoehtoja. Tänä aikana nousi esille kolme erittäin mielenkiintoista tutkimusta, joita tässä työssä hyödynnetään. Näihin kuuluvat ETH Zürichin tutkimusryhmän, johon kuuluvat Philipp Krüsi, Paul Furgale, Michael Bosse ja Roland Siegwart, artikkeli pistepilvien hyödyntämisestä liikesuunnittelussa, Erik Nelsonin avoimen lähdekoodin samanaikainen paikannus- ja kartoitusalgoritmi BLAM ja Neal Seegmillerin pyörillä liikkuvien robottien dynamiikkaohjelmisto. Syksy ja talvi 2017 kuluivat pääasiassa edellisten algoritmien ymmärtämiseen, testaamiseen, muokkaamiseen ja replikoimiseen. Kevään 2018 aikana työn painopiste siirtyi kirjoittamiseen, jota tosin helpotti se, että kirjoitusprosessi oli jo aloitettu viime kesänä. Kesän 2018 aikana viimeistelin työn kirjallisen osuuden.

Haluaisin esittää kiitokset Sandvikille, joka mahdollisti tämän työn tekemisen ja antoi minulle kohtuullisen vapaat kädet sen toteuttamiseen. Sandvikilta haluaisin kiittää erityisesti seuraavia henkilöitä: ohjaajaani Teemu Parkkista, Esa Vikmania, Jussi Puuraa ja Tomi von Esseniä. Haluan myös kiittää tarkastajiani professori Robert Pichéä ja professori Reza Ghabcheloota. Kiitokset myös ystävilleni ja tutuille, jotka ovat antaneet hyviä vinkkejä, neuvoja tai tukea. Lopuksi haluaisin kiittää perhettäni loistavasta tuesta, kannustuksesta ja kasvatuksesta, jotka ovat mahdollistaneet sen, että olen päässyt tähän pisteeseen saakka. Erityiskiitoksen haluan antaa veljelleni Joel Simeon Hyypälle mahtavasta pilkunviilauksesta.

Tampereella, 14.10.2018

Samuel Hyyppä

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

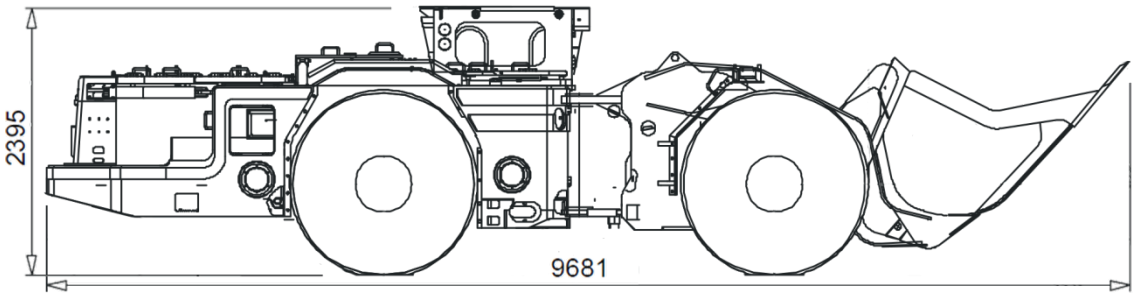
A*	engl. A star, reitinetsintään verkosta käytetty algoritmi, Dijkstran algoritmista nopeampi versio käyttämällä heuristiikkaa
AGV	engl. Autonomous Guided Vehicle, itsenäisesti toimiva ajoneuvo
BLAM	engl. Berkeley Localization and Mapping, SLAM-algoritmi
D*	engl. Dynamic A*, D star, inkrementaalinen heuristinen etsintäalgoritmi
D* Lite	Yksinkertaisempi etsintäalgoritmi kuin alkuperäinen D*, perustuu LPA*:een
DAE	engl. Differential Algebraic Equation, differentiaalinen algebrallinen yhtälö
DPC	engl. Driving on Point Clouds, Krüsi et. al artikkeli [21]
DynamicSWSF–FP	engl. Dynamics Strict Weakly Superior Function – Fixed Point
EKF	engl. Extended Kalman Filter, laajennettu Kalman-suodin
ETH Zürich	saks. Eidgenössische Technische Hochschule Zürich, teknillinen yliopisto Zürichissä, Sveitsissä
FOG	engl. Fibre Optic Gyroscope, kuituoptinen gyroskooppi
Hybrid-A*	A*:een perustuva reitinetsintäalgoritmi, jossa hyödynnetään myös jatkuvia tiloja
ICP	engl. Iterative Closest Point, iteratiivisesti lähin piste
IMU	engl. Inertial Measurement Unit, inertiamittausyksikkö
LHD	engl. Load-Haul-Dump, LHD-lastauskone, kauhakuormaaja
Lidar	engl. Light Detection and Ranging, lasertutka
LPA*	engl. Lifelong Planning A*, perustuu A*:een ja DynamicSWSF–FP:hen
LUT	engl. Lookup Table, etsintätaulu
MATLAB	engl. Matrix Laboratory, numeeriseen laskentaan tarkoitettu ohjelmisto ja ohjelmointikieli
MEMS	engl. Microelectromechanical systems, mikrosysteemit
PCA	engl. Principal Component Analysis, pääkomponenttianalyysi
PCL	engl. Point Cloud Library, avoimen lähdekoodin kirjasto pistepilville ja 3D-geometrialle
RLG	engl. Ring Laser Gyroscope, rengaslasergyroskooppi
ROS	engl. Robot Operating System, joustava ohjelmistokehys robottien ohjelmointiin
RRT	engl. Rapidly-exploring Random Tree, nopeasti tutkiva satunnaispuu
RRT*	Nopeasti tutkiva satunnaispuu -algoritmi, joka konvergoituu kohti optimaalista ratkaisua
SCKF	engl. Square root Cubature Kalman Filter
SLAM	engl. Simultaneous Localization and Mapping, samanaikainen paikannus ja kartoitus
SVD	engl. Singular Value Decomposition, pääakselihajotelma
Theta*	A*:een perustuva reitinsuunnittelualgoritmi, joka voi hyödyntää useita kulmia
VLP-16	Velodyne LiDAR Puck, Velodynen 16-kanavainen reaaliaikainen 3D-lasertutka
WMR	engl. Wheeled Mobile Robot, pyörillä liikkuva robotti

WMRDE	engl. WMR Dynamics Engine, pyörillä liikkuvien robottien dynamiikkaohjelmisto
$\boldsymbol{X}$	Matriisi
$\boldsymbol{x}$	Vektori
$\hat{\boldsymbol{x}}$	Yksikkövektori
$\bar{\boldsymbol{x}}$	Keskiarvovektori

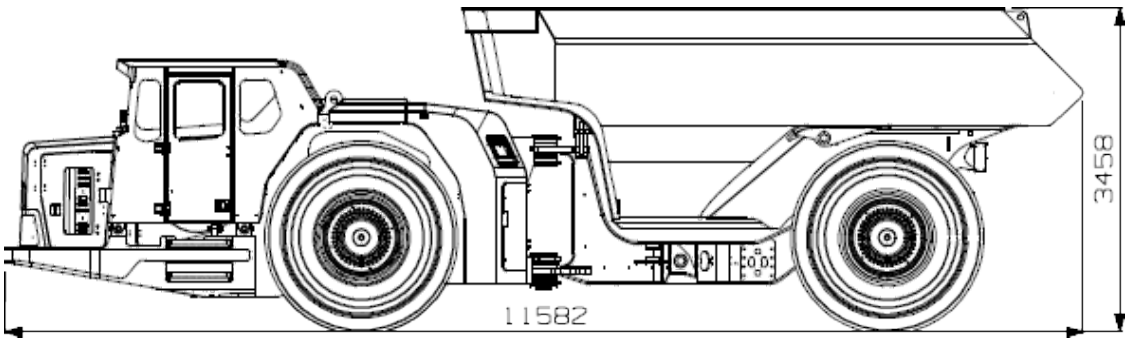
1. JOHDANTO

Kaivoskoneiden automatisointiin on monia syitä. Työtehtävät ovat paljon itseänsä toistavia. Työympäristöt voivat olla vaarallisia sortumavaaran vuoksi tai epämukavia suuren kosteuden ja lämpötilan takia. Lisäksi kuljettajan työmatka maan pinnalta kaivoksen pohjalle voi viedä huomattavasti aikaa. Automatisoinnin avulla yksi operaattori voi hallita useaa ajoneuvoa maanpinnalta käsin. Tällöin kaivoksen tuottavuus parantuu huomattavasti. Automatisoinnin avulla myös työkoneiden kestävyys voi parantua.

Maanalaisissa kaivoksissa malmin kuljettamiseen käytetään tyypillisesti runko-ohjattavia ajoneuvoja. Kuvan 1 LHD-lastauskone (voidaan kutsua myös kauhakuormaajaksi, englanniksi LHD, Load-Haul-Dump) [39] ja kuvan 2 siirtokuormuri [41] ovat runko-ohjattavia ajoneuvoja, jotka koostuvat kahdesta toisiinsa keskinivelellä liitetystä osasta.



Kuva 1. *Runko-ohjattavat ajoneuvot koostuvat kahdesta osasta, jotka yhdistetään toisiinsa keskinivelellä. Etu- ja taka-akseleiden etäisyydet keskiniveleen ovat parempien ajo-ominaisuuksien takia yhtä suuret. Kuvassa on LHD-lastauskone. Yksiköt ovat millimetreinä. Muokattu lähteestä [39].*



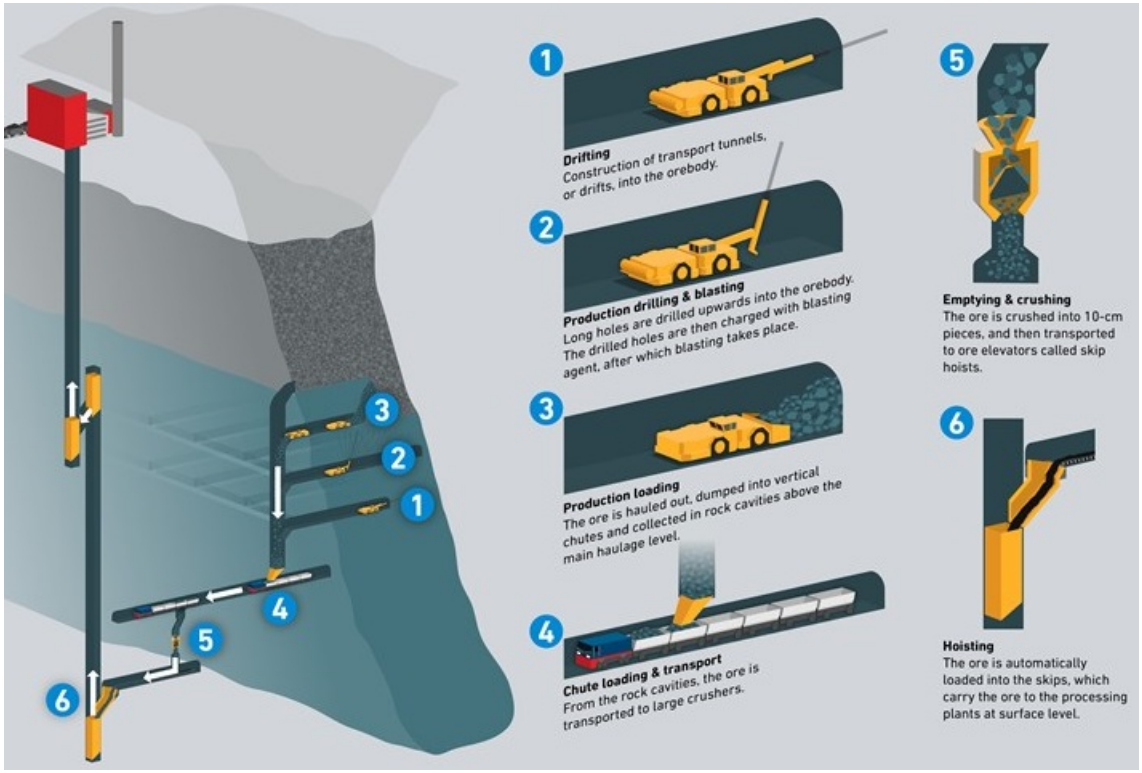
Kuva 2. *Kuvassa on siirtokuormuri (dumpperi). Yksiköt ovat millimetreinä. Muokattu lähteestä [41].*

Kauhakuormaajan pituus voi olla yli 10 m, paino yli 50 000 kg ja kuljetuskapasiteetti yli 20 000 kg. Siirtokuormurin pituus voi olla yli 10 m, paino yli 40 000 kg ja kuljetuskapasiteetti yli 60 000 kg. Altafini käsittelee artikkelissaan [1] runko-ohjattavien ajoneuvojen etuja verrattuna perinteisiin automaisiin ajoneuvoihin ahtaissa kaivostunneleissa. Runko-ohjattavat ajoneuvoilla on paremmat ajo-ominaisuudet etenkin silloin, kun nii-

den etu- ja taka-akselin etäisyydet keskiniveleen ovat yhtä suuret. Tällöin akseleiden keskipisteiden reitit kulkevat käännyttäessä lähempänä toisiaan ja ajoneuvo tarvitsee vähemmän tilaa kääntyäkseen. Ajoneuvoja ohjataan hydraulisilla sylintereillä, jotka muuttavat keskinivelen kulmaa. Sylinterit ovat tarpeeksi voimakkaita, jotta ajoneuvoa voi ohjata myös sen ollessa paikallaan.

Kaivoksessa voi olla yksi tai useampi lastauspiste ja tyhjennyspaikka kaivoksen koosta riippuen. Yksittäinen lastauspiste tai tyhjennyspaikka voi olla käytössä muutamasta päivästä useampaan kuukauteen. Lastauspisteen ja tyhjennyspaikan välimatka voi vaihdella 200 metristä aina 5 kilometriin saakka. Näiden välillä voi olla myös useita reittimahdollisuuksia. Ympäristönä kaivos on hyvinkin dynaaminen toisin kuin aluksi voisi luulla. Esimerkiksi kaivoksen seinät voivat liikkua malmion liikkeen takia, tienpinta voi kulua tai urautua, kaivoksen kosteusolosuhteet voivat muuttua, tiellä voi olla tippuneita kiviä tai tunneleissa voi liikkua muita ajoneuvoja tai mahdollisesti ihmisiä. Lisäksi esimerkiksi siirtokuormurit saattavat liikkua myös tunnelin ulkopuolella. Tällaiset dynaamiset tekijät ympäristössä aiheuttavat haasteita kaivoskoneen navigoinnille.

Kuvassa 3 on esimerkki maanalaisesta louhinnasta [26]. Ensimmäinen vaihe on peränajo, jossa rakennetaan kuljetustunnelit eli perät malmioon. Tuotantoporauksessa perissä porataan pitkät reiät ylöspäin räjähteitä varten. Kun räjäytyksen aiheuttamat kaasut ja pöly on tuuletettu pois, voidaan siirtyä kolmanteen vaiheeseen. LHD-lastauskone käy hakemassa malmia perästä ja kuljettaa sen kaivoksessa eteenpäin.

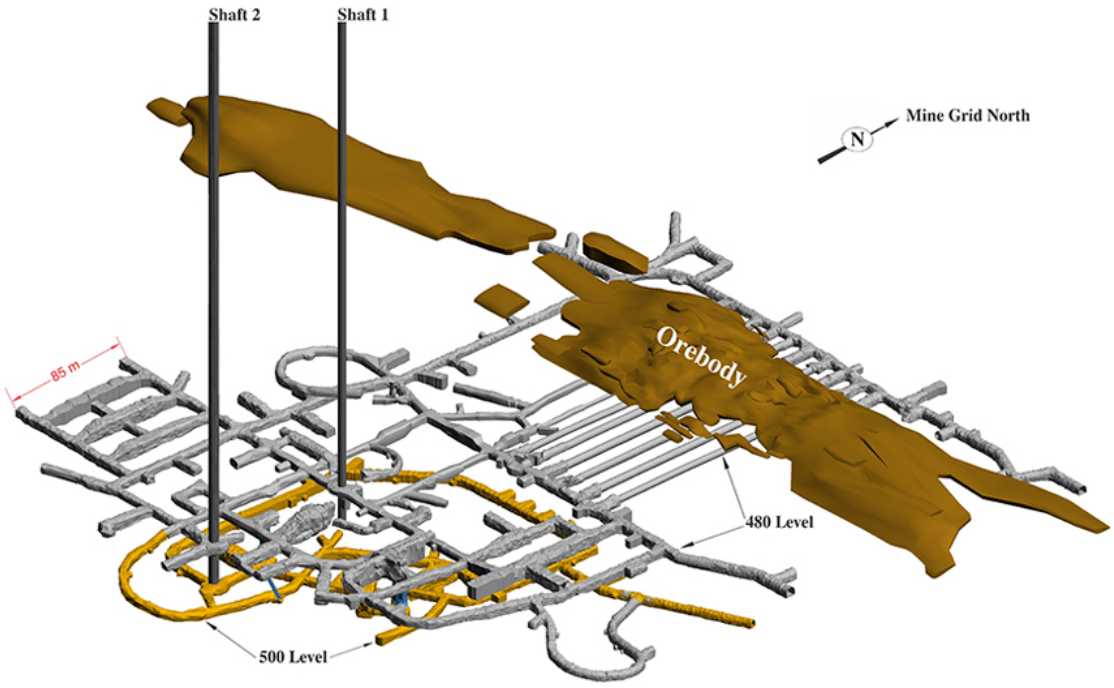


Kuva 3. *Esimerkki levysorroslouhinnan vaiheista: peränajo, tuotantoporaus ja räjäytys, lastaus, rännilastaus ja kuljetus, tyhjennys ja murskaus sekä nosto. [26]*

Edellisen kuvan esimerkissä malmi toimitetaan maanpinnalle junan, murskaimen ja nostolaitteen avulla. Malmi voidaan toimittaa eteenpäin myös siirtokuormureiden avulla, kuten kuvassa 4 havainnollistetaan [14]. Kuvassa 5 on esitetty esimerkki kaivoksen profiilista [7].

Kuva 4. *LHD-lastauskone lastaa malmin kuljetusta varten siirtokuormuriin. [14]*

Kaivoskoneiden automatisointia on tutkittu 1970-luvulta saakka [2]. Navigoinnin automatisointiratkaisuissa on käytetty ympäristöön rakennettavaa lisäinfrastruktuuria, joka on lähtöisin 1960–80-lukujen ensimmäisistä itsenäisesti toimivista ajoneuvoista (AGV, Autonomous Guided Vehicles). Navigoinnissa käytetään tällöin esimerkiksi induktiivisia haudattuja johtoja [15], maalattuja viivoja [8], heijastavia teippejä [19], valoa lähettäviä naruja tai muita paikannusmajakoita. Tällainen ratkaisu saattaa toimia tehdasympäristössä, jossa reitit voivat pysyä samoina useita vuosia. Kuitenkin kaivosympäristössä reitit muuttuvat useammin, jolloin navigointirakenteiden siirtäminen ja lisääminen tulisi liian työlääksi ja kalliiksi pitkällä aikavälillä.



Kuva 5. *Kaivoksen tunnelisto voi olla monimutkainen. [7]*

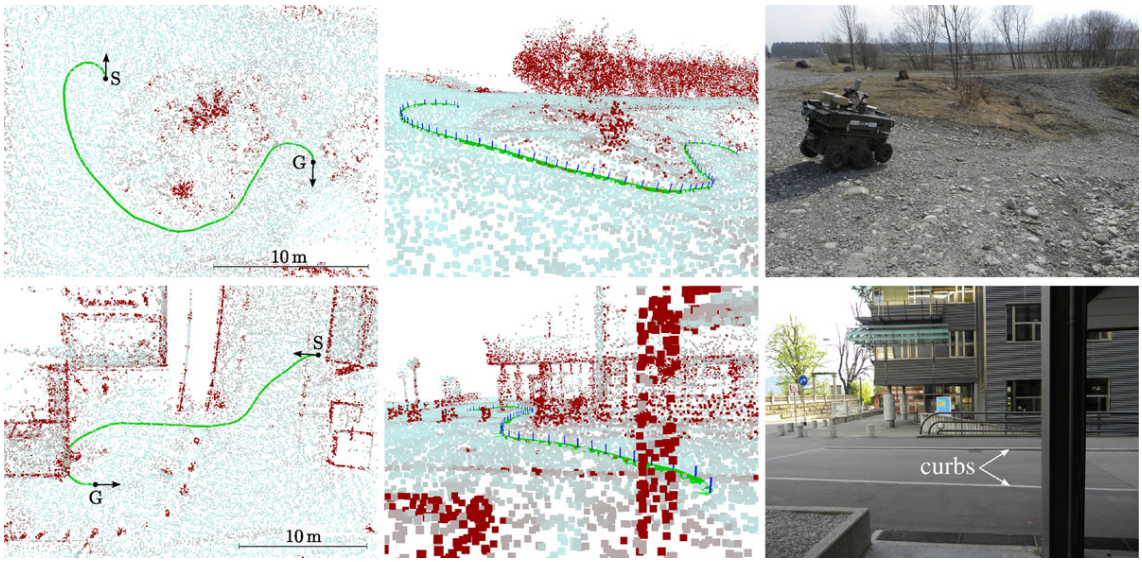
Teollisuudessa onkin jo käytössä sellaisia ratkaisuja, jotka eivät vaadi erillisiä navigointirakenteita tunneleihin. Yksi ratkaisu perustuu pitkälti Mäkelän 1990-luvun aikana tehtyihin tutkimuksiin [27, 28]. Navigointi perustuu ympäristön tallentamiseen, reitin opettamiseen ja lasertutkiin, joiden avulla korjataan arvioidun sijainnin ajautumaa. Kyseissä

menetelmässä ympäristön ja reitin opetus vie kuitenkin huomattavan paljon aikaa. Tämän työn yksi tarkoitus onkin tutkia vaihtoehtoja tulevaisuuden järjestelmälle, joka voisi lyhentää uuden reitin käyttöönottoaikaa.

Mäkelän työssä on käytetty ympäristön opetukseen yksinkertaistettua versiota samanaikaisesta paikannuksesta ja kartoituksesta (SLAM, Simultaneous Localization and Mapping). Smith ja Cheeseman [45] tutkivat paikan epävarmuuden kuvaamista ja estimoimista SLAM-ongelmassa jo vuonna 1986. Myös 1990-luvun alussa Leonard ja Durrant-Whyte [25] tekivät merkittävää tutkimusta SLAM-ongelmasta ja esittivät, että epävarmuutta on mahdollista pienentää pitkällä aikavälillä.

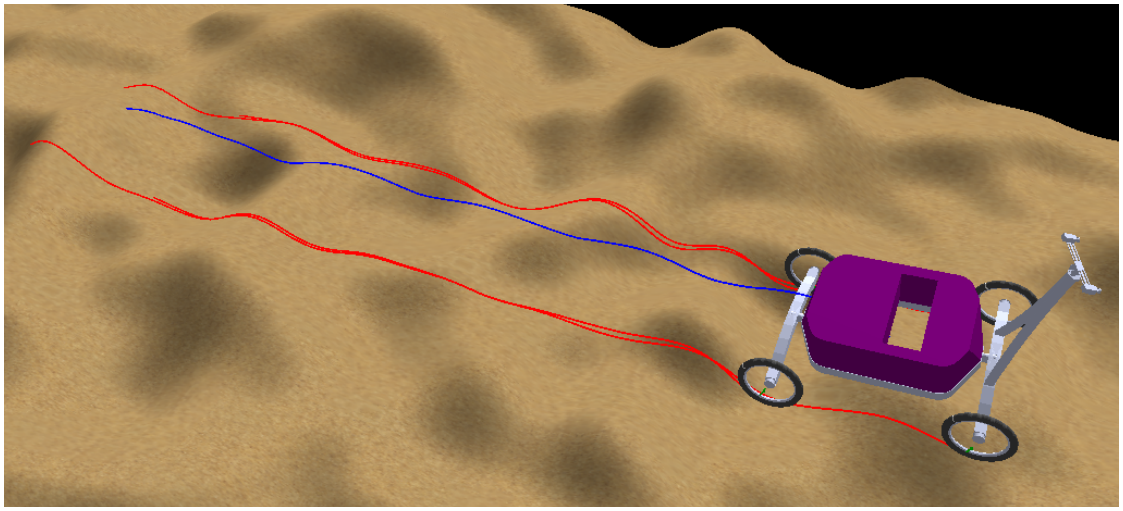
SLAM on yhä aktiivinen tutkimuksen kohde, ja vuosien saatossa siitä on tehty useita algoritmeja, joista osa on myös saatavissa avoimena lähdekoodina. SLAM-algoritmit tehdään kuitenkin hyvin sovelluskohtaisesti. Esimerkiksi ympäristöt, rakennettavat kartat, käytettävissä olevat anturit ja laskentaresurssit poikkeavat toisistaan hyvinkin paljon eri sovellusten välillä. Käytettävät anturit määrittävät hyvinkin paljon SLAM-algoritmin toimintaa. Yleisesti käytettäviä antureita ympäristön havainnointiin ovat kamerat, kaikuluotaimet, tutkat ja lasertutkat (Lidar, Light Detection and Ranging). Kamerat voidaan jakaa vielä RGB-, stereo- ja RGB-D-kameroihin. Näistä viimeisissä on myös syvyysmittaus mukana. Hyvien aikaisempien kokemuksien takia tässä työssä keskitytään lasertutkiin.

ETH Zürichin tutkimusryhmä, johon kuuluu Krüsi, Furgale, Bosse ja Siegwart, esittää artikkelissaan [21] käytännöllisen lähestymistavan maarobottien liikesuunnittelulle ja maaston arviointiin 3D-ympäristöissä. Heidän työhönsä viitataan jatkossa lyhenteellä DPC (Driving on Point Clouds). Heidän järjestelmänsä luo 3D-pistepilven lasertutkan, IMU:n (Inertial Measurement Unit, inertiamittausyksikkö), odometrian ja iteratiivisesti lähimmän pisteen algoritmin (Iterative Closest Point, ICP) avulla. Järjestelmä laskee tietyin parametrein optimoidun 6D-reitin robotille suoraan pistepilveen huomioiden kaarevuus- ja jatkuvuusrajoitteet ilman erillistä pintojen laskentaa, diskretointia tai topologian luontia. Lisäksi järjestelmä toimii dynaamisissa ympäristöissä. Tämä järjestelmä voisi tietyin muokkauksin sopia myös kaivoskoneiden navigointiin. Valitettavasti heidän koodistaan vain osa on avointa, mutta hyvän artikkelin ansiosta se on kuitenkin replikoitavissa. Suurimmat muutokset tulisi tehdä liikesuunnittelulle, jossa tarvitsee huomioida ainakin ajoneuvojen eri kokoluokat. Lisäksi ajoneuvojen kinematiikkamallit poikkeavat toisistaan suuresti ja erityishuomiota vaatii myös keskinivel ja sen muutokset, kun analysoidaan maastoa. Kaivokset ovat kuitenkin varsin rajattuja ympäristöjä, minkä takia jonkinlainen topologian määritys voisi olla hyödyllinen niissä nopeuttamaan liikesuunnittelua. Kuvassa 6 on DPC-tutkimusryhmän järjestelmän tuottama reitti ja pistepilvi kahdesta eri perspektiivistä kahdessa eri ympäristössä [21].



Kuva 6. *DPC-tutkimusryhmän järjestelmän tuottama reitti on esitetty pistepilvessä kahdesta eri perspektiivistä. Lisäksi oikealla on esitetty kuvat oikeista ympäristöistä. [21]*

Jos tarvitaan tarkempaa ajoratojen mallinnusta, voidaan esimerkiksi hyödyntää Seegmillerin väitöskirjassaan [43] esittämää pyörillä liikkuvien robottien dynaamisten mallien muotoilua ja kalibrointia. Kuvassa 7 on esitetty tilannekuva Zoë-maasturin simulaatiosta epätasaisessa 2,5D-maastossa.



Kuva 7. *Tilannekuva Zoë-maasturin simulaatiosta epätasaisessa 2,5D-maastossa. [43]*

Simulaatiota pystyy ajamaan jopa 10 tuhatta kertaa nopeammin verrattuna reaaliaikaan. Tämän vuoksi simulaatiota voi mahdollisesti hyödyntää liikesuunnittelussa tai törmäyksien estämiseen ilman, että ympäristöä joudutaan luokittelemaan. Yhdistettynä kaivostunnelin topologian tai alikarttojen verkoston kanssa voidaan mahdollisesti luoda nopeasti reittejä, joita voidaan käyttää myös myöhemmin.

1.1 Tavoitteet

Tämän työn tarkoitus on hahmotella suuntaviivoja tulevaisuuden kaivoskoneen navigointijärjestelmälle, joka kykenee täyttämään seuraavat tavoitteet:

1. Operaattoreiden työn tulisi vähentyä uusien reittien käyttöönoton ja opetuksen osalta.
2. Järjestelmän tulisi kyetä ratkaisemaan törmäyksetön reitti annettujen pisteiden välille ilman, että operaattorin tarvitsee muokata reittiä erillisillä ohjelmilla.
3. Järjestelmän tulisi käyttää uudelleen jo laskettuja tietoja, kuten vanhoja reittejä ja ympäristömalleja.

Järjestelmän käytettävyyden puolesta olisi hyvä, jos se kykenisi täyttämään myös seuraavat lisätavoitteet:

1. Järjestelmän tulisi kyetä toimimaan myös ympäristöissä, jotka sisältävät dynaamisia kohteita, kuten muita ajoneuvoja.
2. Järjestelmän tulisi kyetä suunnittelemaan reittiään siten, että kaarrokset olisivat mahdollisimman loivia, jotta renkaat eivät kuluisi. Mahdollisesti järjestelmän tulisi kyetä myös varioimaan reittiään hieman, jotta tienpinta ei urautuisi.
3. Toimintaympäristöjä ei välttämättä tarvitsisi rajata vain kaivostunneleihin.

Käytettävissä olevat anturit, jotka määräytyvät joko ympäristökestävyyden, saatavuuden tai kustannusten perusteella, käyttöjärjestelmä ja ajotietokoneen laskentakapasiteetti ovat kaivoskoneen navigointijärjestelmän merkittävimpiä rajoittavia tekijöitä.

1.2 Työn rajaukset

Tekoäly ja koneoppiminen voisivat olla myös eräs ratkaisu kaivoskoneen navigointiin. Viime vuosina on tehty erittäin paljon tutkimusta aiheesta. Googlen DeepMind on kyennyt päihittämään ihmisen jo kiinalaisessa go-lautapelissä, jossa mahdollisten siirtojen määrä on niin suuri, ettei niiden analyyttinen laskeminen ole tehokasta, kuten esimerkiksi shakissa. Pelissä intuitio ja kokemus ovat suuressa roolissa. Lisäksi Elon Muskin rahoittama OpenAI on päihittänyt parhaat ihmispelaajat Dota 2 -tietokonepelissä. Se oppi ennustamaan mihin toinen pelaaja tulee liikkumaan, improvisoimaan tuntemattomissa tilanteissa ja kuinka vaikuttaa vastustajan liittoutuneisiin yksiköihin muutaman kuukauden aikana pelaamalla itseään vastaan. Kuitenkin muutamat pelaajat löysivät taktiikoita joiden avulla he pystyivät päihittämään OpenAI:n avulla luodun botin.

Lähestyttäessä kaivoskoneiden navigaatiota tekoälyn avulla pitäisi esimerkiksi kaivoskoneen toiminta simulaattorissa pelillistää eli esimerkiksi antaa pisteitä suotavasta toiminnasta, kuten määrätyn kohteen saavuttamisesta riittävän nopeasti, ja vähentää pisteitä ei-suotavasta toiminnasta, kuten törmäämisestä esteisiin, ylimääräisestä jarruttamises-

ta tai kiihdyttämisestä, vaarallisessa ympäristössä ajamisesta ja turhista ohjausliikkeistä. Tämän jälkeen tekoäly pyrkii maksimoimaan pistemäärä erilaisissa simulaatiotilanteissa. Lisäksi tekoälyn tulisi huomioida eri ajoneuvojen ominaisuudet ja se, kuinka dynamiikka muuttuu esimerkiksi ajoneuvon ollessa täydessä lastissa. Ongelmana voi olla kuitenkin se, ettei oikea ajoneuvo tule käyttäytymään oikeassa ympäristössä välttämättä samalla tavalla kuin simulaattorissa, jos malleissa on epätarkkuuksia. Tässä työssä ei kuitenkaan keskitytä tekoälyn tuomiin mahdollisuuksiin kaivoskoneen navigointiratkaisuna. En alkanut tutkia tässä työssä tekoälyä ensinnäkin siitä syystä, että olin perehtynyt jo aiemmin Krüsin et al. DPC-menetelmään [21], joka vaikutti erittäin toimivalta ratkaisulta.

1.3 Työn rakenne

Luvussa kaksi esitellään nykyisen järjestelmän toiminta, järjestelmän kehityskohteita, itsenäisen navigaatiojärjestelmän osatekijöitä ja työhön liittyvää teoriaa. Nykyinen järjestelmä perustuu pitkälti Mäkelän työhön [27, 28]. Järjestelmässä käytetyt 2D-lasertutkat aiheuttavat rajoituksia sen toimintamahdollisuuksiin, minkä takia tässä työssä keskitytään 3D-lasertutkiin. Lisäksi luvussa tarkastellaan itsenäisen navigointijärjestelmän tärkeimpiä osatekijöitä, joita ovat kartoitus, paikannus, liikesuunnittelu ja ohjaus. Teoriaosuudessa käsitellään lyhimmän reitin ongelmaa verkkoteorian kautta, pistepilvien yhdistämisessä käytettyä iteratiivisesti lähimmän pisteen algoritmia (ICP), reitinsuunnittelussa käytettyä nopeasti tutkivaa satunnaispuuta (RRT), liikeradoissa käytettyjä kaarevuuspolynomeja (CP) ja pinnan normaalin paikalliseen estimointiin pistepilvestä käytettyä pääkomponenttianalyysiä (PCA).

Luvussa kolme esitellään kaivoskoneen antureita, käytettäviä algoritmeja ja muokkausehdotuksia näihin. Kaivoskoneen antureista käsitellään navigaation kannalta tärkeimmät nykyiset anturit ja hieman myös mahdollisesti lähitulevaisuudessa tulevia antureita, jotka saattavat olla hyödyllisiä ja edullisempia kuin nykyiset. Tässä työssä käytettävät algoritmit ovat DPC-menetelmässä [21] esitelty pistepilviin perustuva navigointialgoritmi, jota käytetään tämän työn pohjana, Nelsonin kehittämä avoimen lähdekoodin reaaliaikaiseen 3D-paikannukseen ja kartoitukseen tarkoitettu ohjelmistopaketti BLAM [34] ja Seegmillerin väitöskirjassaan [31] esittämä pyörillä liikkuvien robottien dynaamisten mallien muotoilu ja kalibrointi.

Luvussa neljä esitellään edellisten algoritmien tuottamia tuloksia. Aluksi tarkastellaan BLAM-algoritmin avulla muodostettua 3D-pistepilveä, johon on käytetty Sandvikin testikaivoksesta kerättyä dataa. Saatua mallia verrataan kaivoksen kiintopisteisiin. DPC-menetelmän liikesuunnittelun algoritmi on muokattu kaivoskoneelle sopivaksi. Koodi on toteutettu MATLAB-ympäristössä. Lopuksi esitellään kaivoskoneen malli ja simulaatio 2,5D-ympäristössä hyödyntäen Seegmillerin työtä. Lopuksi luvussa viisi käydään läpi työn päätulokset ja jatkotutkimusehdotukset.

2. TEOREETTINEN TAUSTA JA KIRJALLISUUS

Tässä luvussa tarkastellaan ensin nykyisen järjestelmän toimintaa ja sen puutteita. Sen jälkeen käydään läpi, mistä osista yleisesti ottaen itsenäinen navigaatiojärjestelmä koostuu. Lopuksi luvussa esitellään työssä käytettäviä teoreettisia menetelmiä.

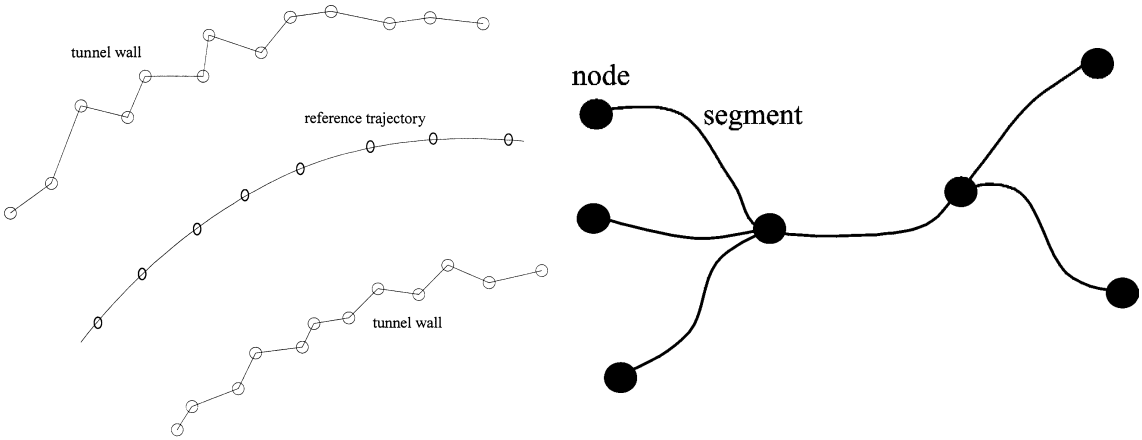
2.1 Nykyisen navigaatiojärjestelmän toiminta ja puutteet

Sandvikin AutoMine®-tuoteperhe kattaa nykyisin automaation kaikki osa-alueet yksittäisen ajoneuvon kauko-ohjauksesta tai autonomisesta toiminnasta usean ajoneuvon ohjaukseen ja koko kaluston täyden automaation hyödyntäen automaattisia tehtäviä ja liikenteenohjausta [3]. Koska nykyinen järjestelmä huolehtii liikenteenohjauksesta, ei ajoneuvojen navigaatiojärjestelmän itse tarvitse olla tietoinen mahdollisista liikennesäännöistä ja muusta liikenteestä, vaan ne voivat keskittyä suorittamaan niille annettua navigaatiotehtävää. Tämän vuoksi työssä keskitytään yksittäisen ajoneuvon navigointiin.

Nykyisin käytössä oleva yksittäisen ajoneuvon navigaatiojärjestelmä perustuu Mäkelän [27, 28] työhön. Käytännössä järjestelmässä on muutamia ongelmia, joista yksi on uuden reitin käyttöönottoon kuluva pitkä aika. Uuden lyhyenkin reitin käyttöönotto voi viedä helposti useita työtunteja. Menettely tapahtuu seuraavasti:

1. Jos ympäristö on uusi, pitää ympäristöstä luoda ensiksi 2D-kartta ajamalla kyseisellä kaivoskoneella ja tallentamalla data lasertutkalla. Jo tunnetussa ympäristössä tämä vaihe voidaan jättää väliin.
2. Uuden reitin tallentaminen aloitetaan tunnetulta alueelta. Ajoneuvoa ohjaa kokenut kuljettaja, joka pystyy huomioimaan tien ja tunnelin paikalliset olosuhteet sekä ohjaamaan puomia ja kauhaa tilanteeseen sopivasti. Opetuksesta tallennetaan reittidata kyseisestä ympäristöstä sekä puomin ja kauhan toiminta.
3. Uusi tallennettu reitti varmistetaan seuraamalla tätä automaattiajolla kahdesti: ensiksi puolella nopeudella ja sitten täydellä nopeudella.
4. Uusi reitti yhdistetään vanhoihin erillisellä ohjelmalla.
5. Lopulliset reitit varmistetaan vielä kerran ennen käyttöönottoa tuotannossa.

Kuvassa 8 on esitetty ympäristömalli ja reitti sekä se, miten reitit voidaan jakaa solmuiksi ja segmenteiksi. Tallennettaessa uutta reittiä tämän päät ovat solmuja ja keskiosa on segmentti. Reittien solmut voivat esimerkiksi olla lastauspisteitä, tyhjennyspisteitä tai risteyksiä. Reitin voi jakaa myös useaan segmenttiin, jos se on tarpeellista. Yhdistämällä tietyt solmut ja segmentit voidaan muodostaa kokonainen reitti eri tunneliston osien välille.



Kuva 8. *Vasemmalla on esitetty tunnelin tallennettu ympäristömalli ja reitti. Oikealla reitit on jaettu solmuiksi ja segmenteiksi. [27]*

Muita ongelmia nykyiselle järjestelmälle aiheuttavat tulevaisuuden järjestelmän vaatimukset. Jos järjestelmän tulisi kyetä laskemaan tietyin parametrein optimaalinen reitti kahden pisteen välillä ja mahdollisesti parantamaan ajan kuluessa tätä reittiä tai muuttamaan sitä, jottei tienpinta kulu tai uraudu liikaa yhdestä kohdasta, käytössä olevilla horisontaalisilla 2D-lasertutkilla tämä ei onnistu ilman riskiä. Nämä lasertutkat eivät voi havaita tason ylä- tai alapuolella olevia kohteita, jolloin reitin muuttaminen pelkästään näiden avulla voi aiheuttaa törmäyksiä. Näiden lasertutkien avulla muodostettu ympäristömalli riippuu lisäksi niiden korkeudesta. Jos ajoneuvojen lasertutkat ovat eri korkeudella, ne tarvitsevat eri ympäristömallit. Lisäksi haasteita muodostavat rampit, joiden vaikutuksesta lasertutkadatassa ajoneuvon edessä tai takana näyttäisi olevan seinä. Nämä seinät näyttävät häviävän, kun ajoneuvo lähestyy ramppia ja sen orientaatio muuttuu rampin normaalin mukaisesti.

Nykyinen järjestelmä tallentaa vain muutaman pisteen sekä oikealta että vasemmalta ympäristömalliin, jotta esimerkiksi rampit eivät aiheuta virheitä ympäristömalliin. Paikantamisessa vertaillaan tallennettua ympäristömallia ja mitattua seinäprofiilia, jotta ajautumat paikassa ja suunnassa saadaan korjattua. Jos tielle on esimerkiksi pudonnut kivi, jonka takia ajoneuvo kallistuu, paikannuksessa saattaa syntyä virhettä. Lisäksi tienpinnan urautuminen tai muu kuluminen voi muuttaa lasertutkan havaintokorkeutta, jolloin paikannus ei välttämättä enää onnistu. Havaintokorkeus voi muuttua myös lastauksen yhteydessä kuorman lisääntymisen vuoksi. 3D-ympäristömallilla ajoneuvon kallistumat tai lasertutkan havaintokorkeuden muutokset eivät aiheuta ongelmaa paikantamisessa. Lisäksi eri ajoneuvot voisivat käyttää samaa tallennettua ympäristömallia. Myös ympäristön samankaltaisuudesta aiheutuvat paikannusvirheet vähenevät.

3D-ympäristömalli mahdollistaa paremman ajoneuvon liikesuunnittelun. Silloin voidaan huomioida liikesuunnittelussa tien paikalliset ominaisuudet, kuten rampin jyrkkyys, tien kallistuma, tien epätasaisuudet ja erilaiset esteet. Kuitenkaan aivan kaikkea ei 3D-lasertutkilla pysty määrittämään, kuten tienpintojen kitkakertoimia ja tien koostumusta, kuten mutaa, jonka takia tienpinnassa tapahtuu mahdollisesti muodonmuutoksia sen yli

ajettaessa. Kirkkaat pinnat, kuten vesi, voivat aiheuttaa ongelmia. Jos tien yli kulkee peilikirkas tasainen vesilätäkkö, se voi näyttää lasertutkadatassa mahdollisesti rotkolta.

Nykyinen järjestelmä ei päivitä ympäristömalliaan itsenäisesti, minkä takia suuret ympäristön muutokset, kuten malmion liikkeen takia siirtyneet seinät, uudet tunnelit tai uudet staattiset esteet ympäristössä, voivat pysäyttää järjestelmän toiminnan, kun paikannus ei ole enää riittävän tarkka. Tulevien järjestelmien tulisi pystyä päivittämään ainakin pienet muutokset ympäristömalliin. Lisäksi niiden tulisi pystyä havaitsemaan mahdolliset dynaamiset kohteet, kuten muut ajoneuvot, tai uudet staattiset esteet, jotka voivat aiheuttaa mahdollisen törmäyksen. Kun mahdollinen törmäysvaara on havaittu, ajoneuvon tulisi ensimmäiseksi pyrkiä estämään mahdollinen törmäys joko suunnittelemalla väistöliike tai pysähtymällä. Jos liikkeen uudelleensuunnittelu ei onnistu edes pysähdyksen jälkeen, ajoneuvo voi ilmoittaa ongelmasta teleoperaattorille jollain keinolla, johon tässä työssä ei oteta kantaa.

Lisäksi olisi suotavaa, että osia uudesta teknologiasta pystyttäisiin käyttämään myös vanhassa kalustossa mahdollisimman vähillä päivityskustannuksilla. Lopuksi vielä yhteenvetona nykyisen järjestelmän ongelmat, joihin tämä työ pyrkii löytämään ratkaisut:

1. 2D-lasertutkat eivät pysty havaitsemaan esteitä havainnointitasonsa ala- tai yläpuolelta, minkä takia automaattinen reitinlaskenta tai mukauttaminen tienpinnan kulumisen estämiseksi eivät ole riskittömiä.
2. Uuden reitin käyttöönotto vie opetuksen takia paljon aikaa.
3. Eri ajoneuvot vaativat mahdollisesti omat ympäristömallinsa.
4. Paikannus voi hukata tarkan sijainnin useissa eri tilanteissa. Esimerkiksi tielle tippuneet kivet voivat kallistaa ajoneuvoa, ajoneuvon renkaiden kuluminen muuttaa anturin havainnointikorkeutta tai uudet staattiset objektit ympäristössä voivat aiheuttaa liian suuria muutoksia mittauksiin, jolloin tarkkaa paikkaa ei pystytä määrittämään tallennetulla ympäristömallilla.
5. Dynaamisissa ympäristöissä järjestelmää ei voi käyttää.

2.2 Itsenäinen navigaatiojärjestelmä

Minkä tahansa todellisesti autonomisen järjestelmän tai robotin tulee kyetä tekemään päätöksiä. Autonomisen järjestelmän tulee kyetä havainnoimaan ympäristöään, tekemään päätöksiä havaintojen sekä sille asetettujen tehtävien perusteella ja toimimaan ympäristössä tehtävien suorittamiseksi. Autonomisten järjestelmien tulisi kyetä toimimaan pitkiäkin aikoja ilman toisten suoraa vaikuttamista. Tällöin järjestelmän pitäisi myös huoltaa tai käydä huollattamassa itseään. Tässä työssä keskitytään kuitenkin vain navigointiin.

Itsenäinen navigaatiojärjestelmä voidaan jakaa useaan osaan, joihin kuuluu ainakin ympäristön havainnointi, kartoitus, paikantaminen, liikesuunnittelu, ohjaus ja turvallisuus.

Järjestelmän täytyy kyetä havainnoimaan ympäristöään, jotta se voisi muodostaa kartan, jota se voi käyttää itsensä paikantamiseen ja myöhemmin liikesuunnitteluun. Suunnittelun luomat ohjeet pitää suorittaa turvallisesti ohjauksen avulla. Itsenäisen navigaation pääaihealueet voidaan jakaa samanaikaiseen paikannukseen ja kartoitukseen (SLAM), liikkumisen suunnitteluun sekä ohjaukseen, joita käsitellään seuraavaksi.

2.2.1 Samanaikainen paikannus ja kartoitus (SLAM)

Yksi olennaisin osa itsenäisiä navigaatiojärjestelmiä on samanaikainen paikannus ja kartoitus (Simultaneous Localization and Mapping, SLAM). SLAM-ongelmissa pyritään muodostamaan ja päivittämään jonkinlaista karttaa tuntemattomasta ympäristöstä ja samalla seuraamaan olion sijaintia siinä. SLAM-algoritmeja on useita, mutta niiden toiminta riippuu käytettävissä olevista resursseista ja käyttötarkoituksesta. Esimerkiksi lattialla toimivan robotti-imurin ei tarvitse yltää ominaisuuksiltaan lähellekään itseajavia autoja.

SLAM-algoritmit suunnitellaan erilaisiin sovelluksiin käytettävissä olevien antureiden ja tarvittavan ympäristöstä muodostettavan kartan perusteella. Erilaisia SLAM-algoritmeissa käytettäviä antureita voivat olla lasertutkat, kamerat, kiihtyvyysanturit, tutkat, kaikuluotaimet, matkamittarit ja ulkoiset paikannusmenetelmät, kuten GPS. Ympäristöstä muodostettava kartta voi olla esimerkiksi 2D-ruutukenttä tai 3D-pistepilvi. 2D-ruudut voivat olla riittäviä esimerkiksi juuri robotti-imureille, ja tarkempaa tietoa ympäristöstä vaativat sovellukset, kuten itseajavat autot, voivat käyttää 3D-pistepilveä.

Monesti SLAM-ongelmaa lähestytään tilastollisesti ajattelemalla suureet todennäköisyyksien avulla. Käyttämällä antureilta saatua diskreettiä havaintodataa SLAM ratkaisee olion sijainnin ja luo karttaa ympäristöstä hyödyntäen Bayesin teoreemaa tai sen approksimaatioita.

Yksi erittäin mielenkiintoinen SLAM-algoritmi on FastSLAM, jonka ovat kehittäneet Montemerlo ja Thrun [30]. Algoritmista on olemassa vanhempi 1.0- ja uudempi 2.0-versio. Uudempi versio on hieman haastavampi implementoida, mutta se pyrkii ottamaan paremmin huomioon tilanteet, joissa robotin liike sisältää suhteellisesti enemmän kohinaa kuin havainnot. FastSLAM-algoritmin ongelmina voivat olla kuitenkin Jacobin matriisien laskenta ja epälineaaristen funktioiden linearisointi. FastSLAM-algoritmista onkin tehty monia muokkauksia, kuten esimerkiksi Zikoksen ja Petridiksen L-SLAM (Low dimensional SLAM) [53]. Hewitt [18] käyttää myös työssään FastSLAM-algoritmia, mutta välttääkseen EKF:n (Extended Kalman Filter) ongelmat hän käyttääkin SCKF:ää (Square root Cubature Kalman Filter).

Hewittin [18] työ on muutenkin hyvin mielenkiintoinen ja aihealueeltaan lähellä myös tätä työtä. Hän käsittelee työssään Mars-mönkijän kaltaista ajoneuvoa, jolle hän esittää 3D-navigointialgoritmin. Hän määrittää ajoneuvolle kinematiikkamallit, joita hyödyn-

netään SLAM-algoritmissa. Lisäksi hän hyödyntää neuroverkkoa ympäristöstä lasertutkan avulla mitatun tai simuloidun datan luokitteluun. Tosin hänen työssään ajoneuvo pysähtyy aina lasertutkan käytön ajaksi, mikä ei sovi tähän työhön.

Ajoneuvojen kinematiikka- tai dynamiikkamallit eivät kuitenkaan ole välttämättömiä SLAM-algoritmien toiminnan kannalta, kuten esimerkiksi Nelson [34] demonstroi BLAM-algoritmillaan (Berkeley Localization and Mapping), jonka hän on julkaissut avoimena lähdekoodina. Hän käyttää vain Velodynen VLP-16-lasertutkasta saatua dataa. Kyseisessä algoritmissa on kuitenkin omat heikkoutensa, minkä vuoksi todelliseen järjestelmään kannattaa soveltaa DPC-menetelmässä [21] esiteltyä SLAM-algoritmia.

2.2.2 Liikesuunnittelu

Liikesuunnittelulla tarkoitetaan robotiikassa lyhyesti ilmaistuna prosessia, jossa haluttu liiketehtävä pilkotaan diskreeteiksi liikkeiksi, jotka noudattavat asetettuja rajoituksia ja mahdollisesti optimoivat kokonaista liikettä jostain näkökulmasta. Liikesuunnittelualgoritmien tulee ottaa huomioon erilaisia rajoitteita ja epävarmuuksia.

Liikesuunnittelualgoritmien käyttökohteina voivat olla esimerkiksi robottien turvallinen navigointi erilaisissa ympäristöissä tai teollisuusrobottien nivelien ohjaus. Samankaltaisia algoritmeja hyödynnetään myös digitaalisten hahmojen animoinnissa, pelien tekoälyssä ja tietokoneavusteisessa suunnittelussa esimerkiksi arkkitehtuurissa ja biologisten molekyylien tutkimuksessa. Luvussa 2.3 käsitellään erilaisia verkkoteoriassa ja navigoinnissa käytettyjä liikesuunnittelualgoritmeja.

2.2.3 Ajoneuvon mallinnus, simulointi ja ohjaus

Tässä työssä ei varsinaisesti keskitytä kaivoskoneen ohjaukseen. Nykyisen laitteiston ohjausjärjestelmän toiminnasta voi lukea Mäkelän työstä [27]. Ajoneuvon ohjaukseen ja tämän suunnitteluun liittyvät vahvasti myös mallinnus ja simulointi.

Kaivoskoneen mallinnusta ja ohjausta käsitellään myös muiden töissä, joista esimerkiksi Dragtin maisterin opinnäytetyö [13]. Siinä hän esittelee erilaisia malleja kaivoskoneille mukaan lukien myös dynaamisia. Lisäksi työssä on esitetty näille simulaatiotuloksia. Hän hyödyntää työssään Lagrangen mekaniikkaa. Tässä työssä sen sijaan on hyödynnetty kaivoskoneiden simulointiin Seegmillerin pyörillä liikkuvien robottien dynamiikkaohjelmisto [43].

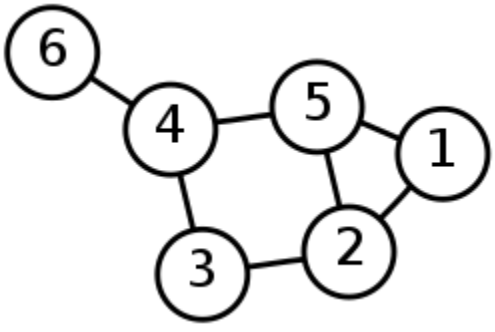
2.3 Lyhimmän reitin ongelma verkkoteoriassa

Dijkstra kehitti 1950-luvun lopulla lyhimmän reitin algoritminsa, jolla voidaan etsiä lyhin reitti kahden solmun välille verkossa. Verkkoteoria sai alkunsa, kun Euler ratkaisi Königsbergin siltaongelman 1700-luvulla. Verkko koostuu solmuista tai pisteistä, joita yhdistetään toisiinsa reiteillä, kaarilla tai reunoilla. Reunat voivat olla myös suunnattuja, eli niillä on tietty suunta solmujen välillä, tai suuntaamattomia, eli kummastakin solmusta pääsee toiseen solmuun. Verkon solmuilla ja reunoilla voi olla erilaisia kustannuksia. Kuvassa 9 on esimerkki yksinkertaisesta verkosta.

DPC-menetelmässä [21], jota käsitellään myöhemmin, liikeradan paikallisessa optimoinnissa käytetään yksinkertaisia verkon etsintäalgoritmeja, kuten Dijkstran algoritmia, parhaan reitin valitsemiseksi. Verkon etsintä algoritmeja käytetään myös reitinsuunnittelussa etenkin 2D-tasossa. Solmut saadaan diskretoimalla tason koordinaatisto esimerkiksi ruuduiksi. Reunoiksi voidaan valita esimerkiksi vierekkäiset 8 ruutua. Dijkstran algoritmi on aika tehoton ratkaisemaan parasta reittiä tasossa, koska se laajentaa solmuja tasaisesti. Vuonna 1968 Hart, Nilsson ja Raphael kehittivät artikkelissaan [17] Dijkstran algoritmia lisäämällä heuristiikan ohjaamaan etsintää. Algoritmia kutsutaan nimellä A^*. Heuristiikan avulla algoritmin tarvitsee käydä läpi vähemmän solmuja, minkä takia se yleensä tuottaa ratkaisun nopeammin kuin alkuperäinen Dijkstran algoritmi.

Muuttuvassa ympäristössä ja reaaliaikaisessa suunnittelussa A^* on huono, koska siinä ei säilytetä lainkaan aiemmin laskettuja tietoja, minkä takia se aloittaa suunnittelun alusta aina muutoksen ympäristössä havaittuaan. 1990-luvun alussa Stentz [48] kehitti tähän ongelmaan dynaamisen A^*- eli D^*-algoritmin, jossa säilytetään aiemmin laskettuja tietoja ja uudelleensuunnittelu on nopeampaa, kun muutos ympäristössä tai verkossa havaitaan. Tästä on myöhemmin kehitetty useita eri versioita. Yksi suosituimmista on Koenigin ja Likhachevin kehittämä D^* Lite, joka nimestään huolimatta ei suoraan perustu alkuperäiseen D^*-algoritmiin vaan heidän aikaisemmin kehittämäänsä LPA^*-algoritmiin (Lifelong Planning A^*), joka taas hyödyntää A^*-algoritmia ja Ramanlingamin ja Repsin [38] kehittämää DynamicSWSF–FP:tä (Strict Weakly Superior Function – Fixed Point). D^* Lite on helpompi ymmärtää ja toteuttaa, ja toiminnallisesti se on vähintään yhtä hyvä kuin alkuperäinen D^* ja siitä kehitetyt versiot.

Useissa tasoreitinsuunnittelualgoritmeissa on ongelmana liikkeiden rajoittuminen diskretoitujen ruutujen naapureihin, minkä takia löydetty reitti ei vastaa todellista optimaalista lyhintä reittiä, joka ei rajoitu ruutuihin. Tähän ongelmaan Nash, Daniel, Koenig ja Felner [32] ovat kehittäneet $Theta^*$-algoritmin, joka pystyy



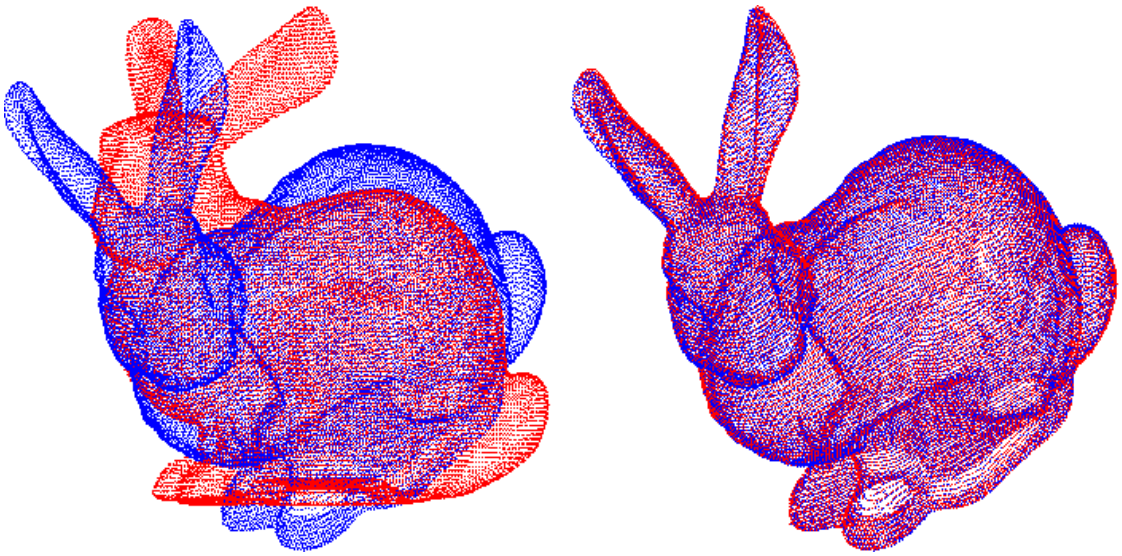
Kuva 9. *Yksinkertainen verkko.*

löytämään lähestulkoon todellisen lyhimmän reitin tasossa.

Kaikissa edellisissä algoritmeissa on kuitenkin se ongelma, että ne tuottavat diskreettejä reittejä, jotka eivät välttämättä ole sulavia ja toteuta jatkuvan maailman ajoneuvon ei-holonomisia rajoituksia (ei-holonomisen systeemin tila riippuu käytetystä reitistä, jolla se on saavutettu, eli se riippuu derivaatoistaan). Tähän ongelmaan Dolgov, Thrun, Montemerlo ja Diebel kehittivät artikkelissaan [12] Hybrid-A^*-algoritmin. Aluksi etsintäavaruus diskretoidaan, kuten A^*-algoritmissakin, ja verkko muodostetaan ruudukosta. Erona A^*-algoritmiin on se, että Hybrid-A^* tallentaa myös ajoneuvon jatkuvan tilan jokaisen ruudun kanssa. Solmuja laajennetaan kolmella eri ohjausvaihtoehdolla (täysi vasen, suoraan tai täysi oikea) ja uudet tilat selvitetään ajoneuvon kinemaattisella mallilla. Jos solmun aikaisempi kustannus on laskettu ja solmun uusi kustannus on pienempi, päivitetään kustannus ja uudelleen priorisoidaan solmu. Algoritmin toisessa vaiheessa löydettyä reittiä optimoidaan. Algoritmia on käytetty DARPA Grand ge -kilpailussa. Hybrid-A^*-algoritmi voisi olla yksi vaihtoehtoinen tapa reitinsuunnitteluun RRT:n sijasta, jota käsitellään myöhemmin.

2.4 Iteratiivisesti lähin piste – ICP

ICP:stä (Iterative Closest Point) eli iteratiivisesti lähimmän pisteen algoritmista on kehitetty useita eri versioita 1990-luvulta lähtien. Näistä ensimmäiset julkaisivat Besl ja McKay [4] sekä Chen ja Medionil [9]. Näiden yksi mahdollinen käyttökohde on esimerkiksi SLAM. Segal, Haehnel ja Thrun [44] käsittelevät työssään ICP-algoritmien perusteita ja lisäksi ehdottavat oman muunnelmansa nimeltä yleistetty ICP. Näiden algoritmien perusideana on pyrkiä löytämään muunnos kahden eri pistepilven välille, kuten kuvassa 10 [36] tehdään. Tämän työssä käsiteltävissä SLAM-algoritmeissa käytetään ICP:tä.



Kuva 10. *ICP:n avulla etsitään muunnos kahden eri pistepilven välille. [36]*

Standardi ICP-algoritmi koostuu kahdesta eri vaiheesta:

1. Ensiksi lasketaan pistepilvien yhtenevyys.
2. Sitten lasketaan muunnos, joka minimoi yhtenevien pisteiden väliset etäisyydet.

Näitä kahta vaihetta toistetaan iteratiivisesti, kunnes tulos konvergoituu haluttuun muunnokseen. Koodille annetaan kaksi pistepilveä $A = \{a_i\}$ ja $B = \{b_i\}$ sekä alkumuunnosmatriisi $\boldsymbol{T}_0$. Koodi tuottaa toivottavasti oikean muunnosmatriisin $\boldsymbol{T}$. Algoritmeille määritellään yleensä maksimisovitusetäisyyden kynnysarvo d_{max}. Tämän avulla määritellään, onko jollekin pisteelle toisessa pistepilvessä yhtenevää pistettä. Arvoa voidaan pitää konvergoituvuuden ja tarkkuuden kompromissina. Ohjelmassa 1 esitetään standardi ICP pseudokoodina.

```
1.  T ← T_0;
2.  while ei-konvergoitunut do
3.  |     for i ← 1 to N do
4.  |     |     m_i ← EtsiLähinPisteAsta(Tb_i);
5.  |     |     if ‖m_i − Tb_i‖ ≤ d_max then
6.  |     |     |     w_i ← 1;
7.  |     |     else
8.  |     |     |     w_i ← 0;
9.  |     T ← argmin_T Σ_i w_i‖m_i − Tb_i‖² ;
```

Ohjelma 1. *Standardi ICP pseudokoodina.*

Chen ja Medionil [9] ovat kehittäneet suositun piste-tasomuunnelman ICP:stä, jossa hyödynnetään myös pisteen $\boldsymbol{m}_i$ paikallista pinnan normaalia $\boldsymbol{n}_i$. Algoritmissa minimoidaankin virheen sijasta pinnan normaalille projisoitua virhettä seuraavasti:

$$\boldsymbol{T} \leftarrow \underset{\boldsymbol{T}}{argmin} \sum_i w_i \|\boldsymbol{n}_i \cdot (\boldsymbol{m}_i - \boldsymbol{T}\boldsymbol{b}_i)\|^2. \tag{1}$$

Yleistetyssä ICP:ssä hyödynnetään molempien pistepilvien paikallisia normaaleja. Kun oletetaan molempien pistepilvien yhtenevien pisteiden noudattavan normaalijakaumia $\boldsymbol{a}_i \sim \mathcal{N}(\hat{\boldsymbol{a}}_i, \boldsymbol{C}_i^A)$ ja $\boldsymbol{b}_i \sim \mathcal{N}(\hat{\boldsymbol{b}}_i, \boldsymbol{C}_i^B)$, voidaan johtaa suurimman uskottavuuden estimoinnin (MLE) avulla minimoitava yhtälö seuraavaan muotoon, jonka erikoistapauksia aikaisemmat muunnelmat ovat:

$$\boldsymbol{T} = \underset{\boldsymbol{T}}{argmin} \sum_i \boldsymbol{d}_i^{(\boldsymbol{T})^T} \left(\boldsymbol{C}_i^B + \boldsymbol{T}\boldsymbol{C}_i^A\boldsymbol{T}^T\right) \boldsymbol{d}_i^{(\boldsymbol{T})}, \qquad \boldsymbol{d}_i^{(\boldsymbol{T})} = \boldsymbol{m}_i - \boldsymbol{T}\boldsymbol{b}_i. \tag{2}$$

Paikallisten normaalien laskemiseen käytetään pääkomponenttianalyysia (PCA), jota käsitellään myöhemmin. Kovarianssimatriisit $\boldsymbol{C}_i^A$ ja $\boldsymbol{C}_i^B$ saadaan asettamalla pisteiden normaalien suuntaiset kovarianssit pieniksi verrattuna pinnan suuntaisiin.

DPC-työssä [21] käytetty ICP perustuu saman laitoksen aikaisempaan tutkimukseen [37]. Pomerleau, Colas, Siegwart ja Magnenat vertailivat siinä eri ICP-muunnelmia ja julkaisivat avoimen lähdekoodin modulaarisen ICP-kirjaston nimeltä *libpointmatcher*.

2.5 Nopeasti tutkiva satunnaispuu – RRT

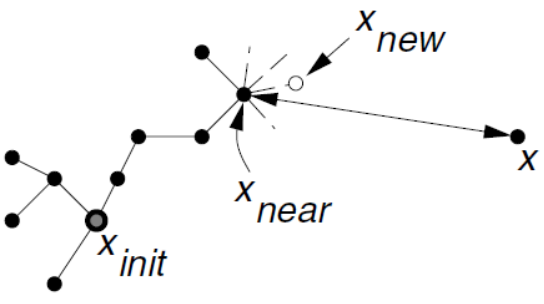
1990-luvun lopussa LaValle esitteli raportissaan [22] nopeasti tutkivan satunnaispuun (Rapidly-exploring Random Tree, RRT) reitinsuunnitteluongelmille. Algoritmista on sen jälkeen tehty useita eri versiota ja niitä käytetään erittäin paljon autonomisessa robotiikassa etenkin liikesuunnittelussa. Yksi suosittu variaatio on Karamanin ja Frazzolin [20] kehittämä RRT*, jonka on osoitettu konvergoituvan kohti optimaalista ratkaisua. Algoritmit etsivät reittiä satunnaispuun avulla ei-konveksista moniulotteisista avaruuksista. RRT:t pystyvät huomioimaan ei-holonomiset ja kinodynaamiset rajoitukset liikesuunnittelussa myös korkean vapausasteen järjestelmille.

RRT-algoritmin pseudokoodi on esitetty algoritmissa 2 mukaillen LaValllen ja Kuffnerin artikkelin [23] esitystapaa. Puu $\mathcal{T}$ alustetaan tilaan x_{init}, minkä jälkeen valitaan satunnainen tila x_{rand}, jonka suuntaan puuta yritetään laajentaa puun lähimmästä pisteestä x_{near}.

```
1.  RakennaRRT(x_init)
2.  |   T.init(x_init);
3.  |   for ei-loppunut do
4.  |   |   x_rand ← SatunnainenTila();
5.  |   |   Laajenna(T, x_rand);
6.  |   return T
7.  Laajenna(T, x)
8.  |   x_near ← LähinNaapuri(T, x);
9.  |   if UusiTila(x, x_near, x_new, u_new) then
10. |   |   T.LisääPiste(x_new);
11. |   |   T.LisääReuna(x_near, x_new, u_new);
12. |   |   if x_new = x then
13. |   |   |   return Saavutettu;
14. |   |   else
15. |   |   |   return Edetty;
16. |   return Ansassa;
```

Ohjelma 2. *RRT pseudokoodina.*

Jos laajentaminen onnistuu, uusi piste x_{new} lisätään puuhun siihen liitetyn ohjauksen u_{new} kanssa. Puun laajentamista jatketaan, kunnes lopetusehto täyttyy. Kuvassa 11 on esitetty puun laajentaminen [23]. Yleensä RRT:tä käytettäessä kasvatetaan kahta eri puuta, joista toinen lähtee alkupisteestä ja toinen loppupisteestä. Näitä kasvatetaan, kunnes niiden yhdistäminen onnistuu.

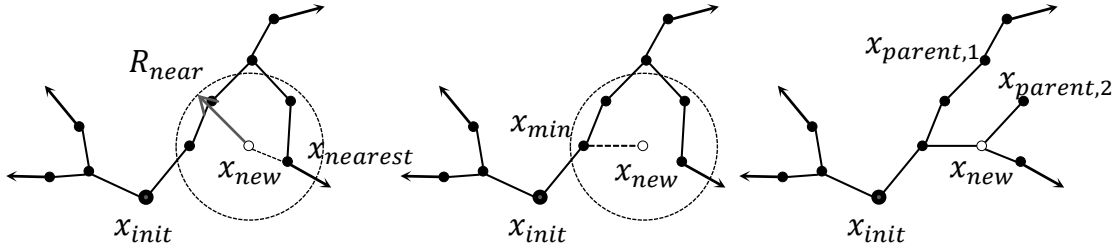


Kuva 11. *RRT:n laajentaminen. [23]*

Karaman ja Frazzol käsittelevät artikkelissaan [20] kattavasti useita eri variaatioita näytteenottoon perustuvista reitinsuunnittelu algoritmeista. Heidän RRT*-algoritmi eroaa normaalista RRT:stä siten, että heidän puunsa pisteitä voidaan johdottaa uudelleen. Kun on valittu satunnainen tila x_{rand}, etsitään puusta tätä lähinnä oleva piste $x_{nearest}$ ja määritetään mahdollinen puuhun lisättävä piste x_{new} käytettävissä olevien ohjauksien perusteella puun lähimmästä pisteestä. Tämän jälkeen etsitään puusta kaikki pisteet X_{near}, jotka ovat tietyn etäisyyden R_{near} päässä tästä uudesta pisteestä. Näistä lähellä olevista pisteistä valitaan se minimikustannuksen tuottava piste x_{min}, jonka kautta uusi piste liitetään puuhun. Seuraavaksi tarkastetaan, voidaanko uuden pisteen kautta johdottaa uudelleen muita lähellä olevia puun pisteitä X_{near}. Uudelleenjohdotus tapahtuu, jos kustannus lähellä olevaan pisteeseen x_{near} on pienempi uuden pisteen kautta kuin sen aikaisempi kustannus. Näiden pisteiden edeltäjäksi vaihdetaan uusi piste. Ohjelmassa 3 on esitetty RRT*-algoritmi pseudokoodina mukaillen Karamanin ja Frazzolin [20] esitystapaansa. Kuvassa 12 on havainnollistettu uudelleenjohdotusta.

1. $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset;$
2. **for** *ei-Loppunut* **do**
3. | $x_{rand} \leftarrow SatunnainenTila();$
4. | $x_{nearest} \leftarrow LähinPistePuusta(\mathcal{T} = (V, E), x_{rand});$
5. | $x_{new} \leftarrow Ohjaus(x_{nearest}, x_{rand});$
6. | **if** $EiEsteitä(x_{nearest}, x_{new})$ **then**
7. | | $X_{near} \leftarrow LähelläOlevatPisteetPuusta(\mathcal{T} = (V, E), x_{new}, JokinEtäisyysF(\dots));$
8. | | $V \leftarrow V \cup \{x_{new}\};$
9. | | $x_{min} \leftarrow x_{nearest}; c_{min} \leftarrow c(x_{nearest}) + c(x_{nearest}, x_{new});$ // Kustannus
10. | | **foreach** $x_{near} \in X_{near}$ **do** // Yhdistetään minimikustannusreittiin
11. | | | **if** $Törmäyksetön(x_{near}, x_{new}) \wedge c(x_{near}) + c(x_{near}, x_{new}) < c_{min}$
12. | | | | **then** $x_{min} \leftarrow x_{near}; c_{min} \leftarrow c(x_{near}) + c(x_{near}, x_{new});$
13. | | $E \leftarrow E \cup \{(x_{min}, x_{new})\};$
14. | | **foreach** $x_{near} \in X_{near}$ **do** // Uudelleen johdotus
15. | | | **if** $Törmäyksetön(x_{new}, x_{near}) \wedge c(x_{new}) + c(x_{new}, x_{near}) < c(x_{near})$
16. | | | | **then** $x_{parent} \leftarrow PisteenEdeltäjäPuussa(x_{near});$
17. | | | | $E \leftarrow \left(E \backslash \{(x_{parent}, x_{near})\}\right) \cup \{(x_{new}, x_{near})\};$
18. **return** $\mathcal{T} = (V, E);$

Ohjelma 3. *RRT* pseudokoodina.*



Kuva 12. *Esimerkki RRT*:n uudelleenjohdotuksesta.*

2.6 Kaarevuuspolynomit

Nagy ja Kelly esittävät [31] konferenssiartikkelissaan kolmannen asteen kaarevuuspolynomit (Cubic Curvature Polynomial, CCP). Käyttämällä vain yhtä tällaista polynomia

voidaan määrittää jatkuva liikerata mihin tahansa loppuasentoon tasossa. Jatkuvuuden ansiosta ajoneuvot pystyvät seuraamaan niitä kohtuullisen helposti, koska ohjauksen vaatima vääntömomentti on myös jatkuva. Ajoneuvo voi kuitenkin aiheuttaa rajoituksia kaarevuuden arvoihin. Ajoneuvon fyysiset ominaisuudet määrittävät ainakin kaarevuuden maksimiarvon. Nämä rajoitukset täytyy ottaa huomioon tarkasteltaessa kaarevuuspolynomin toteuttamiskelpoisuutta. Toinen ongelma kaarevuuspolynomeissa on niiden haastava laskeminen, joka täytyy tehdä numeerisesti.

Nagyn ja Kellyn [31] konferenssiartikkelin mukaan tason asento esitetään sijainnin (x, y), suunnan θ ja kaarevuuden κ avulla seuraavasti:

$$\boldsymbol{x} = [x \quad y \quad \theta \quad \kappa]^T. \tag{3}$$

Artikkelissa on käytetty seuraavaa yksinkertaistettua kinematiikkamallia ajoneuvolle:

$$\begin{cases} \dot{x} = v(t)\cos\theta(t) \\ \dot{y} = v(t)\sin\theta(t) \\ \dot{\theta} = \kappa(t)v(t) \\ \dot{\kappa} = \dot{\alpha}(t)v(t) \end{cases}, \tag{4}$$

missä v on nopeus ja α on ohjauskulma. Todelliset ajoneuvojen mallit ovat monimutkaisempia, mutta käsitellään ensiksi tätä mallia. Mallissa nopeus ei vaikuta liikeradan muotoon, joten liikeradat voidaan mallintaa matkan funktiona. Määritellään kaarevuus matkan funktiona kolmannen asteen polynomin avulla seuraavasti:

$$\kappa(s) = \kappa_0 + as + bs^2 + cs^3, \qquad \kappa_0, a, b, c \in \mathbb{R} \text{ ja s} \in \mathbb{R}_{\geq 0} \tag{5}$$

Suunnan ja paikan funktiot ovat seuraavat:

$$\theta(s) = \theta_0 + \int_0^S \kappa(s) = \theta_0 + \kappa_0 s + \frac{as^2}{2} + \frac{bs^3}{3} + \frac{cs^4}{4} \tag{6}$$

$$x(s) = x_0 + \int_0^S \cos\big(\theta(s)\big), \qquad y(s) = y_0 + \int_0^S \sin\big(\theta(s)\big). \tag{7}$$

Yleispätevyyden vuoksi kannattaa määritellä alkuasennon suunta θ_0 ja sijainti (x_0, y_0) nolliksi eli ainoastaan alkukaarevuutta κ_0 muutetaan. Sijainnin vaatimille integraaleille ei ole olemassa analyyttista ratkaisua, joten asennon ratkaisemiseen joudutaan käyttämään numeerisia menetelmiä. Esimerkiksi MATLAB-ohjelmassa voi käyttää *integral*-työkalua. Trigonometrisia funktioita voidaan myös approksimoida sarjakehitelmien avulla.

Käänteisessä ratkaisussa, jossa pyritään laskemaan kolmannen asteen kaarevuuspolynomin parametrit $[a \quad b \quad c \quad s]^T$ annetun alku- ja loppuasennon perusteella, Nagy ja Kelly käyttävät [31] artikkelissaan alkuarvauksena heuristiikkaa, jossa he olettavat parametrin c olevan nolla, ja heuristisesti määritettyä parametria s, joiden perusteella lasketaan parametrit a ja b. Alkuarvauksena voidaan myös käyttää aikaisemmin laskettuja parametreja, jos aikaisempi asento on lähellä nykyistä. Alkuarvauksen avulla lasketaan asento numeerisia menetelmiä käyttäen ja verrataan tulosta haluttuun asentoon. Jos tarkkuuden ehdot täyttyvät, voidaan iterointi lopettaa. Muuten approksimoidaan Jacobin matriisi käyttämällä lähellä sijaitsevaa loppuasentoa. Saatu matriisi invertoidaan ja kerrotaan virheellä. Saadut muutokset skaalataan alaspäin numeerisen stabiiliuden takia ja lisätään nykyisiin parametreihin.

Artikkelissa on mainittu laskentaa nopeuttavia keinoja, kuten hakutaulukot (Lookup Table, LUT). Siinä ei ole kuitenkaan mainittu, että parametrit ovat skaalattavissa etäisyyden mukaan siten, että käyrän muoto säilyy samana skaalasta riippumatta. Toisin sanottuna suunta on sama suhteellisen matkan ollessa sama. Kun esimerkiksi skaalataan etäisyydet kertoimella n, saadaan uudeksi loppusijainniksi kohta (nx_f, ny_f), uusiksi alku- ja loppukaarevuuksiksi $\frac{\kappa_0}{n}$ ja $\frac{\kappa_f}{n}$ sekä kaarevuuspolynomin pituudeksi ns_f. Tällöin voidaan ratkaista uudet kaarevuuspolynomin parametrit $[a_n \quad b_n \quad c_n \quad s_n]^T$ vanhojen parametrien $[a \quad b \quad c \quad s]^T$ avulla seuraavasti:

$$\theta(s_n) = \theta(s), \qquad s_n = ns$$
$$\kappa_{0,n} = \frac{k_0}{n}, \qquad a_n = \frac{a}{n^2}, \qquad b_n = \frac{b}{n^3}, \qquad c_n = \frac{c}{n^4}, \qquad s_{f,n} = ns_f. \tag{8}$$

Käytettäessä skaalausta riittää, että laskee hakutaulukon arvot vain tietylle etäisyydelle ja että skaalaa nämä halutulle etäisyydelle, jolloin saadaan vastaavan muotoinen skaalattu reitti. Lisäksi voidaan hyödyntää symmetriaa, jonka ansiosta voidaan laskea vain toisen puolen arvot. Vaihtamalla parametrien a, b ja c sekä alkukaarevuuden κ_0 merkit saadaan x-akselin suhteen peilattu käyrä.

Kaivoskoneiden eli yleensä runko-ohjattavien ajoneuvojen kinematiikkamalleista on useita eri versioita [13, 33, 42]. Nämä perustuvat pitkälti Schedingin [42] esittämään laskentatapaan, ja tässäkin työssä käytetään pohjana hänen esitysmuotoaan kinematiikkamallista renkaiden liukukulmien (slip angle) kanssa. Oletuksena on kuitenkin, että liukukulmat ovat suuruudeltaan nollia tai vähintään pieniä. Tämä on tosin huono oletus runko-ohjattavien ajoneuvojen kanssa, mutta liikesuunnittelun helpottamiseksi käytetään tätä oletusta ja jätetään virheiden korjaus ajoneuvon ohjausjärjestelmälle.

Jos oletetaan liukukulmat pieniksi, voidaan takaosan suunnan muutokselle $\dot{\theta}_2$ johtaa seuraava kaava mukaillen Schedingin et al. [42] esitystapaa:

$$\dot{\theta}_2 = \frac{v_2 \sin(\alpha) - L\dot{\alpha}}{L(\cos(\alpha) + 1)}, \tag{9}$$

missä α on keskinivelen ohjauskulma ja L on keskinivelen ja akseleiden välinen etäisyys. Käyttäen kaarevuuden ja ohjauskulman välistä yhteyttä

$$\kappa = \frac{\tan\frac{\alpha}{2}}{L}, \qquad \alpha = 2\,\mathrm{atan}(L\kappa)\,, \qquad \dot{\alpha} = \frac{2L}{L^2\kappa^2 + 1}\dot{\kappa} \tag{10}$$

ja trigonometriaa saadaan johdettua seuraava yhtälö:

$$\begin{aligned} \dot{\theta}_2 &= \frac{v\sin(\alpha)}{L(\cos(\alpha)+1)} - \frac{\dot{\alpha}}{\cos(\alpha)+1} \\ &= \frac{v\tan\left(\frac{\alpha}{2}\right)}{L} - \frac{2L\dot{\kappa}}{L^2\kappa^2+1}\frac{1+L^2\kappa^2}{2} = v\kappa - L\dot{\kappa} \end{aligned} \tag{11}$$

Muodostetaan suunta etäisyyden funktiona vastaavasti kuin aikaisemmin. Ensiksi kirjoitetaan aikaderivaatta auki ja kerrotaan dt:llä, minkä jälkeen jaetaan vielä ds:llä, jolloin saadaan seuraava:

$$\frac{d\theta_2}{dt} = \frac{ds}{dt}\kappa - L\frac{d\kappa}{dt} \rightarrow \frac{d\theta_2}{ds} = \kappa - L\frac{d\kappa}{ds}. \tag{12}$$

Kaarevuus matkan funktiona voidaan ratkaista myös Laplace-muunnosten avulla, jos tiedetään suunnan funktio. Tosin suunnan funktioita ei välttämättä ole helposti saatavilla. Lisäksi kaavasta (12) havaitaan, että suunnan derivaatta on jatkuva, kun kaarevuuden derivaatta on jatkuva. Tämän takia vain kolmannen asteen kaarevuuspolynomi ei ole riittävä. Tarvitaan vähintään neljättä astetta oleva kaarevuuspolynomi. Tällöin aikaisemmat kaarevuuden, takaosan suunnan ja paikan sekä kaarevuuden derivaatan yhtälöt ovat seuraavat:

$$\kappa(s) = \kappa_0 + as + bs^2 + \cdots + n_k s^k, \qquad \kappa_0, a, \ldots, n_k \in \mathbb{R}, k \in \mathbb{N}, \mathrm{s} \in \mathbb{R}_{\geq 0} \tag{13}$$

$$\dot{\kappa}(s) = a + 2bs + \cdots + kn_k s^{k-1}, \qquad a = \dot{\kappa}_0 \tag{14}$$

$$\begin{aligned} \theta_2(s) &= \theta_0 + \int_0^S \kappa(s) - L\int_0^S \dot{\kappa}(s) \\ &= \theta_0 + \kappa_0 s + \frac{\dot{\kappa}_0 s^2}{2} + \frac{bs^3}{3} + \cdots + \frac{n_k s^{k+1}}{k+1} - L(\dot{\kappa}_0 s + bs^2 + \cdots + n_k s^k), \end{aligned} \tag{15}$$

$$x_2(s) = x_0 + \int_0^S \cos\big(\theta(s)\big), \qquad y_2(s) = y_0 + \int_0^S \sin\big(\theta(s)\big). \tag{16}$$

Valitettavasti runko-ohjattavan ajoneuvon kinematiikkamallin takia kaarevuuspolynomit eivät enää ole skaalattavissa kuten aikaisemmassa yksinkertaisessa mallissa. Käänteisen ongelman eli kaarevuuspolynomin parametrien määrittämisen voi ratkaista myös käyttämällä olemassa olevia epälineaarisia optimointimenetelmiä. Tässä työssä käytetään kaksivaiheista ratkaisua, jonka ensimmäisessä vaiheessa optimoitava funktio on asetettu vain nollaksi ja käytetään pelkkiä rajoituksia ja jonka mahdollisessa toisessa vaiheessa käytetään neliöllistä kustannusfunktiota. Esimerkiksi MATLAB-ympäristössä voidaan käyttää *fmincon*-työkalua. Epälineaarinen optimointitehtävä formuloidaan seuraavasti:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \; siten, että \begin{cases} \boldsymbol{A} \cdot \boldsymbol{x} \leq \boldsymbol{b} \\ \boldsymbol{A}_{eq} \cdot \boldsymbol{x} = \boldsymbol{b}_{eq} \\ \boldsymbol{c}(\boldsymbol{x}) \leq 0 \\ \boldsymbol{c}_{eq}(\boldsymbol{x}) = 0 \\ \boldsymbol{l}_{bound} \leq \boldsymbol{x} \leq \boldsymbol{u}_{bound} \end{cases}, \tag{17}$$

missä f on minimoitava funktio, $\boldsymbol{x}$ on ratkaisuvektori, $\boldsymbol{A}$ on lineaarinen epäyhtälörajoitematriisi, $\boldsymbol{A}_{eq}$ on lineaarinen yhtäsuuruusrajoitematriisi, $\boldsymbol{c}(\boldsymbol{x})$ on epälineaarinen epäyhtälörajoitematriisi, $\boldsymbol{c}_{eq}(\boldsymbol{x})$ on epälineaarinen yhtäsuuruusrajoitematriisi ja $\boldsymbol{l}_{bound}$ sekä $\boldsymbol{u}_{bound}$ ovat ratkaisun ala- ja ylärajavektorit.

Kaarevuuspolynomin voi ratkaista asettamalla epälineaariset yhtäsuuruusrajoitteet siten, että paikka, suunta, kaarevuus ja sen derivaatta ovat halutut matkan loppupisteessä s_f seuraavasti:

$$\boldsymbol{c}_{eq}(\boldsymbol{x}) = \begin{cases} x_2(s_f) - x_f \\ y_2(s_f) - y_f \\ \theta_2(s_f) - \theta_f \\ \kappa(s_f) - \kappa_f \\ \dot{\kappa}(s_f) - \dot{\kappa}_f \end{cases} = 0. \tag{18}$$

Jos alkusuunta on normalisoitu nollaksi ja halutaan välttää silmukoiden syntyminen reittiin, voidaan rajoittaa esimerkiksi suunnan maksimiarvoa seuraavasti:

$$c_1(\boldsymbol{x}) = \max_s |\theta_2(s)| - 2\pi \leq 0, \qquad 0 \leq s \leq s_f. \tag{19}$$

Jotta saadaan liikerata kaarevuuden puolesta kuljettavaksi, voidaan lisäksi rajoittaa myös kaarevuuden maksimiarvoa seuraavasti:

$$c_2(\boldsymbol{x}) = \max_s |\kappa(s)| - \kappa_{max} \leq 0, \qquad 0 \leq s \leq s_f. \tag{20}$$

Tämän ehdon takia ratkaisua ei välttämättä löydy esimerkiksi silloin, kun loppupiste sijaitsee lähellä alkupistettä, mutta se on huomattavasti sivussa nykyiseen kulkusuuntaan

nähden. Kaarevuuden derivaattaa voidaan rajoittaa samoin kuin kaarevuuttakin, mutta tämä vaikeuttaa edelleen ratkaisun löytämistä.

Ratkaisun ala- ja ylärajan asettaminen voi olla haastavaa, jos halutaan saada ratkaisu moniin erilaisiin tilanteisiin. Perustilanneessa loppupiste sijaitsee alkupisteen edessä sopivan etäisyyden päässä eikä suunnan muutokset ole liian suuria etäisyyteen verrattuna. Myös kaarevuuden arvojen muutoksien tulisi olla etäisyyden kannalta katsottuna sopivia. Tässä työssä on käytetty parametrien ala- ja ylärajana ± 1:tä, kun parametrit on skaalattu matkan avulla seuraavasti:

$$c_s = cs, d_s = ds^2, \dots, n_{k_s} = n_k s^{k-2}. \tag{21}$$

Hyvin suurella todennäköisyydellä on olemassa parempia skaalauksia ja rajoja erilaisiin tilanteisiin. Etäisyyden minimiarvona käytetään alaspäin skaalattua (75 %) lähdön ja lopun välistä suoraa etäisyyttä. Maksimiarvona käytetään kaaren pituutta ympyrälle, jonka halkaisija on suora lähdön ja lopun välillä, paitsi jos halkaisija on pienempi kuin ennalta asetettu minimiarvo (15 m). Alkuarvauksena voi käyttää liikerataa, jonka pituus on etäisyys lähdön ja lopun välillä ja muut parametrit on asetettu nolliksi. Ensimmäisessä vaiheessa kustannusfunktiona käytetään nollaa, jolloin vain pyritään löytämään jokin ratkaisu, joka täyttää asetetut rajoitukset.

Jos ensimmäinen vaihe tuottaa ratkaisun, voidaan ratkaisua käyttää toisen vaiheen alkuarvauksena ja mahdollisesti tiukentaa ratkaisun ala- ja ylärajoja. Toisessa vaiheessa käytetään seuraavaa neliöllistä kustannusfunktiota:

$$Cost(\boldsymbol{x}) = \int_0^S Q\big(\kappa(s)\big)^2 + R\big(\dot{\kappa}(s)\big)^2. \tag{22}$$

Tässä työssä käytetään kaarevuuden kustannuskertoimena Q arvoa $0{,}1$ ja kaarevuuden muutoksen kustannuskertoimena R arvoa $0{,}9$. Nämä arvot vaikuttavat tuottavan kohtuullisen hyviä reittejä ilman, että ratkaisun löytäminen epäonnistuu.

Kun kaarevuuspolynomin aste on vain neljä, edellä oleva menetelmä konvergoituu monesti toteuttamiskelvottomaan pisteeseen. Käytettäessä viidettä tai sitä suurempaa astetta löydetään yleensä ratkaisu perustilanteisiin, jos ratkaisu on olemassa. Kuitenkin asteiden määrän kasvaessa myös iterointikierrosten määrä yleensä kasvaa. Käytettäessä viidettä astetta haastavillekin tilanteille ensimmäinen vaihe saadaan ratkaistua alle kahdellakymmenellä iteroinnilla. Toinen vaihe ratkeaa yleensä vähemmillä iteroinneilla. Iterointien maksimimäärää (noin 20–50) ei kuitenkaan kannata pitää liian suurena, koska ratkaisua ei rajojen sisältä välttämättä löydy. Tässä työssä käytetään *fmincon*-työkalun sisäpisteoptimointialgoritmia (interior-point algorithm), josta voi lukea lisää lähteistä [5, 6, 52]. Numeerisesta optimoinnista voi lukea lisää esimerkiksi Nocedalin ja Wrightin kirjasta [35].

2.7 Pinnan normaalin estimointi pistepilvestä pääkomponenttianalyysilla

Tässä työssä pääkomponenttianalyysia (Principal Component Analysis, PCA) käytetään paikallisen normaalin määrittämiseen pistepilvistä. Tätä tarvitaan maaston paikalliseen arviointiin pistepilvestä, mitä tarkastellaan myöhemmin tässä työssä. Valitaan esimerkiksi sijainnille $\boldsymbol{t}_{MR}$ lähimmät K pistettä pistepilvestä. Näille pisteille lasketaan massakeskipiste ja kovarianssimatriisi seuraavalla tavalla:

$$\overline{\boldsymbol{p}} = \frac{1}{K} \sum_{k \in \mathcal{N}_{\mathrm{K}}(\boldsymbol{t}_{MR})} \boldsymbol{p}_k \quad \text{ja} \tag{23}$$

$$Cov(\boldsymbol{t}_{MR}) = \sum_{k \in \mathcal{N}_{\mathrm{K}}(\boldsymbol{t}_{MR})} (\boldsymbol{p}_k - \overline{\boldsymbol{p}})(\boldsymbol{p}_k - \overline{\boldsymbol{p}})^T \in \mathbb{R}^{3\mathrm{x}3}. \tag{24}$$

Orientoimaton pinnan normaali saadaan kovarianssimatriisin pienintä ominaisarvoa vastaavasta ominaisvektorista $\boldsymbol{v}_0$. Pinnannormaalin $\boldsymbol{n}_{surf}$ voi orientoida eli suunnata pinnasta ulospäin esimerkiksi tietyn pisteen havainnointisuunnan $\boldsymbol{d}_{ori}$ tai annetun suunnan perusteella, jolloin seuraava epäyhtälö pitää paikkaansa:

$$\boldsymbol{n}_{surf} \cdot \boldsymbol{d}_{ori} > 0. \tag{25}$$

Ominaisvektoreiden laskennan pientä nopeuttamista varten ainakin MATLAB-ohjelmistoa käytettäessä kannattaa ensiksi pisteistä poistaa keskiarvot eli massakeskipiste ja sitten käyttää pääakselihajotelmaa (Singular Value Decomposition, SVD):

$$[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = svd(\boldsymbol{P} - \overline{\boldsymbol{P}}, 'econ'), \qquad \boldsymbol{P} \in \mathbb{R}^{3\mathrm{xK}}, \qquad \overline{\boldsymbol{P}} = repmat(\overline{\boldsymbol{p}}, 1, K) \tag{26}$$

$$\boldsymbol{v}_0 = \boldsymbol{U}(:, end), \qquad \boldsymbol{U} \in \mathbb{R}^{3\mathrm{x}3}, \qquad \boldsymbol{n}_{surf} = sign(\boldsymbol{d}_{ori} \cdot \boldsymbol{v}_0)\boldsymbol{v}_0. \tag{27}$$

Pääakselihajotelmasta voidaan laskea vain matriisin pienintä dimensiota vastaava määrä vasemmanpuoleisia ominaisvektoreita, joita tässä on kolme. Näistä ominaisvektoreista valitaan pienintä ominaisarvoa vastaava vektori. Paikallinen pinnan normaali saadaan orientoimalla edellinen vektori esimerkiksi havainnointivektorin avulla.

3. KAIVOSKONEIDEN ANTURIT JA TYÖN NAVIGOINNISSA KÄYTETTÄVÄT ALGORITMIT

Tässä luvussa esitellään ensiksi kaivoskoneissa mahdollisesti käytettäviä antureita, joihin kuuluu muun muassa asentoanturit, gyroskoopit ja lasertutkat. Antureiden jälkeen käydään läpi työssä käytetyt algoritmit ja mahdollisia muokkausehdotuksia näille. Ne käsittelevät SLAM:ää, liikesuunnittelua ja ajoneuvon simulointia.

3.1 Kaivoskoneissa käytettävät anturit ja vaihtoehdot näille

Kuva 13. *Absoluuttisen optisen asentoanturin tarkkuutta voi parantaa lisäämällä sen levyyn bittejä uusien kierrosten avulla, jolloin erilaisten mahdollisten segmenttien määrä kasvaa. [10]*

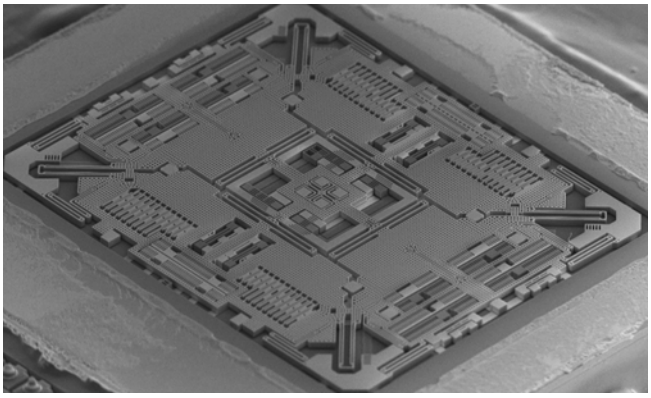
Mäkelän työssä [27] kaivoskoneen antureina käytetään kääntönivelen kulmamittaria, matkamittaria, gyroskooppia ja kahta 2D-lasertutkaa. Kaivoskoneen paikkaa ja suuntaa päivitetään kolmen ensimmäisen anturin avulla käyttämällä navigoinnista tuttuja merkintälaskun (dead reckoning) menetelmiä. Matkamittarin resoluutio on parempi kuin 5 mm ja tarkkuus on yleensä 0,5–2 %:n sisällä. Päivitystaajuus on 25 Hz. Gyroskooppina on käytetty kuituoptista gyroskooppia, jonka päivitystaajuus on 20 Hz ja ajautuma on ±10°:n/h sisällä. Se on asennettu kaivoskoneen takaosaan. Etuosan suuntaa varten tarvitaan myös tieto kääntönivelen asennosta, joka saadaan optisella asentoanturilla, jonka resoluutio on parempi kuin 0,1°. Kuvassa 13 on esimerkki absoluuttisen optisen asentoanturin levystä [10]. Merkintälaskun ajautumaa korjataan kahdella lasertutkalla. Niissä on pyörivät peilit, jotka heijastavat lasersäteen ajoneuvoon nähden horisontaalisessa tasossa 180°:n suuruiselle sektorille. Mittauksia otetaan yhden asteen välein aina lähimpään kohteeseen. Skannauksia otetaan 10 Hz:n taajuudella.

Gyroskoopilla voidaan mitata suunta tai kulmanopeus. Vanhanaikaiset mekaaniset gyroskoopit säilyttävät pyörimismääränsä. Kuvassa 14 on esitetty mekaanisen gyroskoopin perusrakenne [46]. Liikkuvien osien takia mekaaniset gyroskoopit ovat kuitenkin

herkkiä tärinälle, kiihdytyksille ja iskuille. Mekaanisten gyroskooppien valmistuskin on erittäin vaativaa, koska ne tarvitsevat erityiset laakerit ja koska niiden täytyy olla täydellisessä tasapainossa. Akseleiden kitka aiheuttaa myös ajautumaa. Lisäksi mekaaniset gyroskoopit voivat kärsiä kardaanikehyksen lukkiutumisesta (gimbal lock), jolloin menetetään vapausaste. Perinteisessä kolmiakselisessa gyroskoopissa tämä tarkoittaa sitä, että akseleista kaksi on samansuuntaisia, jolloin pyörintä voi tapahtua hetkellisesti vain kahden akselin ympäri. Mekaanisille gyroskoopeille on kuitenkin olemassa useita vaihtoehtoja, kuten esimerkiksi pienelektroniikassa suositut MEMS-gyroskoopit (Microelectromechanical systems, mikrosysteemit) sekä kuituoptiset gyroskoopit (FOG, fibre optic gyroscope) ja rengaslasergyroskoopit (RLG, ring laser gyroscope).

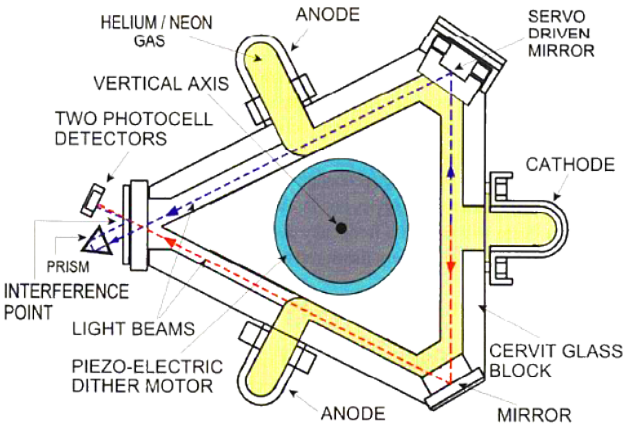
Kuva 14. *Mekaaninen gyroskooppi koostuu usein kolmesta kardaanikehyksestä. Lisäämällä neljäs aktiivisesti ajettu kardaanikehys voidaan välttää kehysten lukkiutuminen. Muokattu lähteestä [46].*

MEMS-gyroskoopit voivat olla erittäin halpoja, ja niitä on helposti saatavilla. Niitä voidaan valmistaa samanlaisilla perustekniikoilla kuin muitakin puolijohdelaitteita, kuten erilaisilla kerrostuksilla, kuvioinneilla ja poistoilla. Ne perustuvat värähteleviin rakenteisiin. MEMS-gyroskoopit voidaan liittää samaan yksikköön kiihtyvyysantureiden kanssa. Kuvassa 15 on esimerkki MEMS-gyroskoopin rakenteesta [49]. MEMS-gyroskoopit ovat suorituskyvyltään kuitenkin heikompia kuin lasergyroskoopit, joiden ajautumat ovat paljon pienempiä.



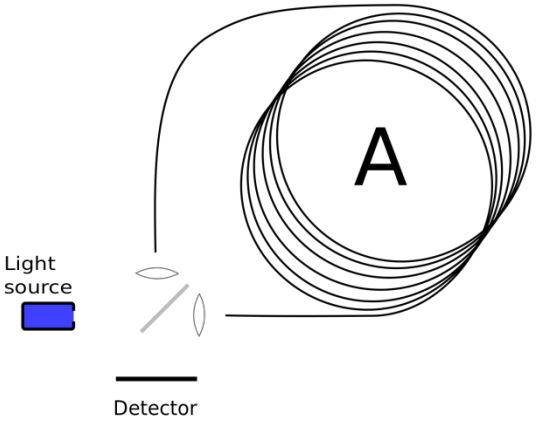
Kuva 15. *ST L3G3250A on kolmiakselinen MEMS-gyroskooppi. [49]*

Lasergyroskooppien toiminta perustuu Sagnac-ilmiöön. Ilmiön mukaan kahdelta valon säteeltä kuluu hieman eri aika kulkiessaan vastakkaisiin suuntiin samaa reittiä, kun reitin sisältämä kappale pyörii minkä tahansa sen tasoa kohtisuorassa olevan



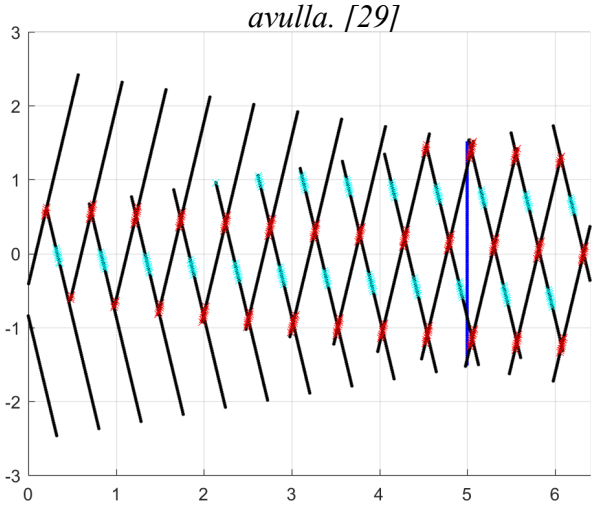
Kuva 16. *Rengaslasergyroskoopin optisessa resonaattorissa etenee samaa reittiä vastakkaisiin suuntiin kaksi lasersädettä, joiden taajuuksien eroa käytetään pyörimisliikkeen havaitsemiseen. Muokattu lähteestä [16].*

akselin ympäri. Lasergyroskooppien toimintaperiaatteet eroavat kuitenkin hiukan toisistaan. RLG:t perustuvat kaasulasereihin suljetussa optisessa rengasresonaattorissa, jossa on erittäin korkealaatuisia peilejä. Pyörimisliike havaitaan taajuuksien eron avulla. FOG:issa valon säde ohjataan vastakkaisiin suuntiin jopa usean kilometrin pituiseen valokuituun ja pyörimisliike havaitaan vaihesiirron avulla. Kuvassa 16 on RLG:n perusrakenne [16] ja kuvassa 17 on FOG:n perusrakenne [29].



Kuva 17. *Kuituoptisen gyroskoopissa valo ohjataan vastakkaisiin suuntiin jopa usean kilometrin pituiseen valokuituun. Pyörimisliike havaitaan vaihesiirron avulla. [29]*

RLG:n katodin ja anodien välille syntyy kaksi lasersädettä, jotka kiertävät eri suuntiin peilien avulla muodostetussa rengasmaisessa optisessa resonaattorissa. Optista resonaattoria voidaan kutsua myös optiseksi kaviteetiksi. Lasersäteet voivat kiertää useita tuhansia kertoja optisessa resonaattorissa muodostaen sen sisälle seisovan aaltoliikkeen. Sagnac-ilmiön takia lasersäteiltä kuluu hiukan eri aika kulkea renkaan ympäri, kun renrengas pyörii. Optisessa resonaattorissa lasersäteen jaksojen määrä pysyy kuitenkin samana, minkä takia lasersäteiden aallonpituudet ja taajuudet muuttuvat hieman. Yksi optisen resonaattorin peileistä on osittain läpäisevä, jonka avulla voidaan tarkastella lasersäteitä resonaattorin ulkopuolella. Lasersäteet ohjataan prisman avulla valodiodeille, joilla tarkastellaan lasersäteiden muodostamaa interferenssikuviota. Taajuuksien ero aiheuttaa interferenssijuovien liikkumisen antureiden yli tällä samalla huojuntataajuudella, joka on myös suoraan verrannollinen pyörimisliikkeeseen. Laskemalla yksittäisten juovien liike anturin yli saadaan vähittäinen pyörimisliike ratkaistua.



Kuva 18. *Kahden eriaallonpituisen lasersäteen interferenssissä juovat liikkuvat huojuntataajuuden mukaisesti joko ylös- tai alaspäin.*

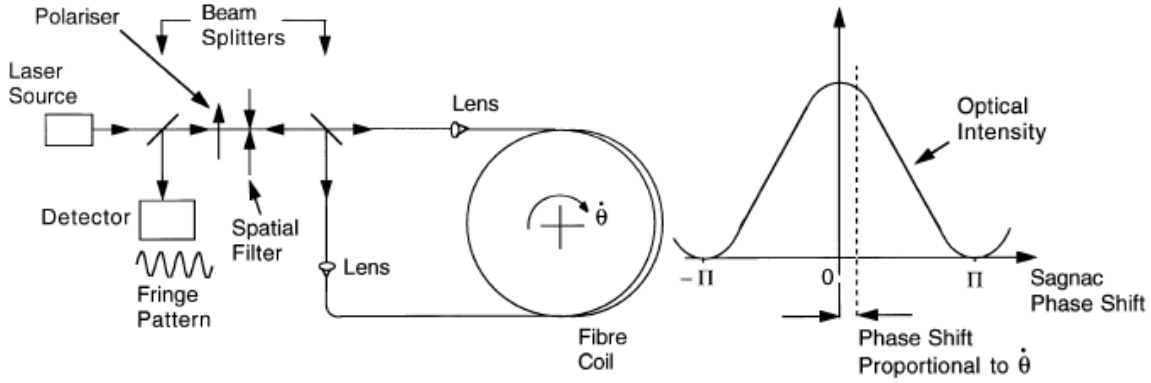
Kuvassa 18 on esimerkki interferenssikuvion juovien liikkeestä, kun ylempää tulevan säteen aallonpituus on suurempi kuin alempaa tulevan. Konstruktiiviset juovat on merkitty punaisella ja dekonstruktiiviset sinivihreällä. Tarkasteltaessa sinisellä anturitasolla juovat liikkuvat ylöspäin.

RLG:t kärsivät kuitenkin pienillä pyörimisnopeuksilla ($< 100°/\mathrm{h}$) lukkiutumisesta (lock-in). Kun vastakkaisiin suuntiin etenevien lasersäteiden taajuus on lähes sama, la-

sersäteet voivat injektiolukkiutua samaan taajuuteen eivätkä enää muutu pyörimisliikkeen mukaan. Interferenssikuvio ei silloin muutu eikä pyörimisliikettä enää havaita. Lukkiutuminen voi johtua esimerkiksi epätäydellisistä peileistä, jotka aiheuttavat takaisinsirontaa lasersäteille, jolloin toisen lasersäteen taajuus alkaa muuttua kohti toista, ja näin molempien säteiden taajuudet siirtyvät kohti samaa taajuutta. Lukkiutuminen estetään optisen resonaattorin usean sadan hertsin taajuisella pienikulmaisella pyörimisvärähtelyllä, jolloin lukkiutuminen tapahtuu hetkellisesti vain värähtelyliikkeen ääripäissä, kun pyörimisnopeus on nolla.

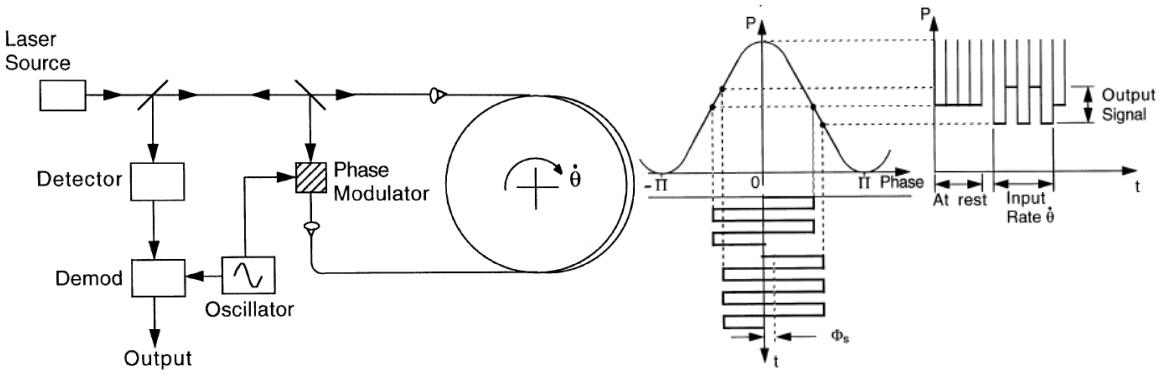
FOG:issa on monia hyötyjä verrattuna RLG:ihin. Niiden elinikä on paljon pidempi, eivätkä ne vaadi ennakoivaa huoltoa, koska niissä ei ole liikkuvia osia, optista resonaattoria, josta väliaine voi vuotaa, tai kuluvia korkean jännitteen elektrodeja. RLG:t vaativat huoltoa ehkä noin viiden vuoden jälkeen. FOG:issa käytettävien optisten kuitujen käyttöiän on tietoliikenneteollisuudessa todettu olevan useita kymmeniä vuosia. Laserdiodienkin käyttöikä on noin parikymmentä vuotta. Viime vuosikymmeninä FOG:t ovat kehittyneet suorituskyvyltään paremmiksi kuin RLG:t.

Yksinkertaisen avoimen silmukan FOG:n, jossa säteet kulkevat symmetrisesti samaa reittiä eri suuntiin, heikkoudet ovat sen herkkyyden puute pienille pyörimisnopeuksille ja ei-symmetriset vaikutukset säteiden reitillä. Säteet ovat täsmälleen samassa vaiheessa, kun kuitukerä on levossa, mutta kerän pyöriessä intensiteetti pienenee hieman pyörimisnopeuden mukaan. Kuvassa 19 on FOG:n perusrakenne ja anturin vaste [11].



Kuva 19. *FOG:n perusrakenne ja anturin vaste [11].*

Herkkyyttä pienille pyörimisnopeuksille voidaan parantaa esimerkiksi lisäämällä kuidun toiseen päähän vaihemodulaattori, joka toimii kuten viivelinja. Molemmat säteet saavat saman vaihemodulaation mutta eri aikoihin. Modulaattori tuottaa $\pm\pi/2$ vaihevärähtelyn, jonka taajuus on asetettu siten, että jakson puolikkaan aikana valo ehtii kulkea kelan läpi. Kuvassa 20 on esitetty vaihemoduloidun avoimen silmukan FOG ja tämän anturin vaste [11].

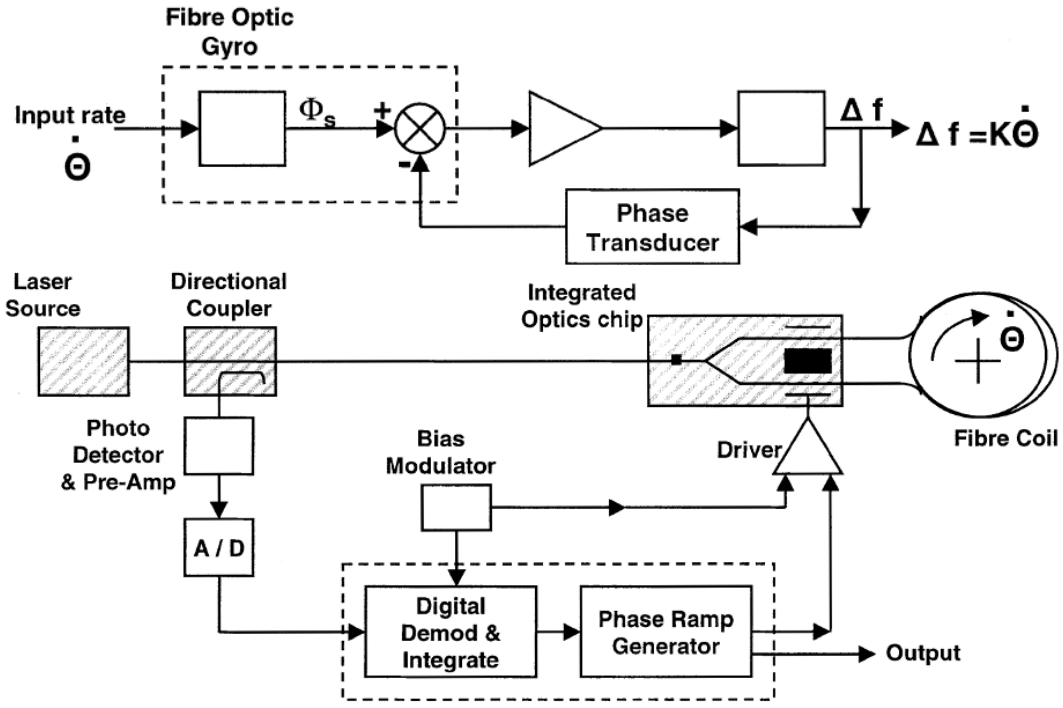


Kuva 20. *Avoimen silmukan FOG:n herkkyyttä voidaan parantaa vaihemodulaattorilla. Oikealla ylhäällä on vaihemoduloimattoman ja alhaalla vaihemoduloidun FOG:n anturin vaste. [11]*

Tällainen avoimen silmukan FOG kärsii kuitenkin vielä ainakin seuraavista kahdesta ongelmasta:

1. Ulostulon stabiilisuus ja suuruusluokkakerroin ovat riippuvaisia demodulaattoria edeltävien vahvistimien arvoista, sen stabiilisuudesta ja laservalon intensiteetin stabiilisuudesta.
2. Suurilla pyörimisnopeuksilla ulostulo on epälineaarinen.

Suljetun silmukan FOG:t pyrkivät joko ohjatun vaiheensiirron tai taajuudensiirron avulla asettamaan sensorin ulostulon haluttuun kohtaan esimerkiksi suurimman herkkyyden alueelle. Koska tämä ohjaussuure on suoraan verrannollinen pyörimisnopeuteen, sen perusteella saadaan FOG:n ulostulon arvo. Kuvassa 21 on esimerkki suljetun silmukan FOG:n ideasta ja rakenteesta [11]. Gyroskoopeista voi lukea lisää lähteistä [11, 24, 50].



Kuva 21. *Suljetun silmukan FOG:n idea ja rakenne [11].*

Navigoinnin kannalta yksi tärkeimmistä antureista on lasertutka (Lidar, Light Detection and Ranging). Lasertutkat ovat aktiivisia antureita, eli ne lähettävät lyhyitä laserpulsseja mitattaviin kohteisiinsa ja mittaavat takaisin heijastuneet pulssit. Hyödyntämällä paluuaikoja ja mahdollisesti muuttuneita aallonpituuksia voidaan määrittää etäisyydet kohteisiin. Lisäksi paluupulssista saadaan selville kohteen kirkkaus. Ensimmäiset lasertutkat kehitettiin pian laserin keksimisen jälkeen 1960-luvulla.

Nykyisin etenkin autoteollisuudessa tutkitaan lasertutkien käyttämistä itseajavissa autoissa. Kuvassa 22 on Velodynen VLS-128-lasertutka [51]. Lasertutkien ongelmana on kuitenkin niiden kallis hinta, joka on tippumisestaan huolimatta yhä useita tuhansia euroja kappaletta kohden. Tämän työn kirjoittamisen aikana eri laitevalmistajat ovat kuitenkin kehittämässä erilaisia keinoja pienentääkseen hintaa. Yksi tutkimussuunta on puolijohdelasertutkat (Solid-State Lidar), joissa ei ole lainkaan liikkuvia osia. Ne käyttävät optisia vaiheistettuja antenniryhmiä (Optical Phased Array), joka muistuttaa tekniikkaa, jota on käytetty myös erilaisissa tutkissa. Valmistajien tavoitteena olisi saada aikaiseksi lasertutkia, joiden yksikköhinta painuisi ainakin muutaman sadan euron luokkaan ja koko mahdollistaisi niiden asentamisen pienempiin paikkoihin kuin nykyisin. Tällöin voitaisiin hyödyntää samassa ajoneuvossa useita lasertutkia laajemman havainnointikentän saavuttamiseksi ja mahdollisesti myös osittain redundanttisuutta vika- tai virhetilanteiden välttämiseksi.

Kuva 22. *Velodynen VLS-128 on 128-kanavainen lasertutka. Aikaisempaan HDL-64-tutkaan verrattuna se on 70 % pienempi ja sisältää kaksinkertaisen määrän kanavia. Lisäksi sen luvattu hinta on pienempi. [51]*

Lasertutkissa käytettävät aallonpituudet ovat yleensä 600–1500 nm:n luokkaa. Koska lasertutkia käytetään ihmisten läheisyydessä, niiden maksimitehoa joudutaan rajaamaan silmien turvallisuuden takia. Tehon rajoittaminen heikentää anturin havainnointietäisyyttä. Havainnointietäisyyteen vaikuttaa kuitenkin hyvin paljon ympäristössä olevien kappaleiden pinnan materiaali, jonka heijastavuus voi vaihdella suuresti. Nykyisin päästään jopa muutaman sadan metrin suuruisiin havainnointietäisyyksiin, mutta heikosti heijastavien materiaalien kohdalla ei saavuteta vastaavia etäisyyksiä. Tietenkin materiaalin ollessa liian heijastava, kuten maanpinnalla oleva vesilätäkkö, voi tulla ongelma, koska lasersäde heijastuu siitä johonkin muualle ja aiheuttaa sen, että lätäkön kohta näyttää melkein rotkolta. Myös kanavien määrällä ja pystysuuntaisen näkökentän koolla, jolle kanavien lasersäteet on jaettu, on paljon merkitystä etenkin esteiden havaitsemiseen yksittäisestä paikasta. Etenkin kapeat horisontaaliset ja joskus myös vertikaaliset kohteet voivat olla

haastavia havaita. Kapea on kuitenkin anturikohtainen käsite, ja liikkeen takia nämäkin kohteet pystytään havaitsemaan.

Lasertutkille voisivat olla vaihtoehtona erilaiset RGB-kamerajärjestelmät. Lasertutkilla on kuitenkin muutama etu verrattuna näihin. Koska lasertutkat ovat aktiivisia antureita, valaistusolosuhteet eivät vaikuta niihin. 3D-kamerajärjestelmät ja kuvien prosessointi vaativat kuitenkin paljon laskentatehoja. Etäisyyden mittaaminen tapahtuu kolmiolaskennalla, mutta sen lisäksi pitää ensiksi varmistaa, että kyseisessä laskutoimituksessa käytettävät kuvapisteet vastaavat toisiaan esimerkiksi erilaisten ominaispiirteiden avulla. Lasertutkien laserpulssien paluuaikoihin perustuva etäisyyden mittaaminen on paljon yksinkertaisempi. Lasertutkissa ja kamerajärjestelmissä laskenta tehdään varta vasten niille suunnitelluilla integroiduilla mikrosiruilla, jolloin säästetään laskentaresursseja kokonaisjärjestelmän muille osille. Kamerajärjestelmien pääedut ovat värien havainnointi ja yksittäisellä hetkellä otetun kuvan tarkempi resoluutio verrattuna yhteen laserskannaukseen. Värien havainnoinnin ansiosta esimerkiksi kylttien tunnistaminen on niille paljon helpompaa riittävissä valaistusolosuhteissa.

Nykyisin käytettävät 2D-lasertutkat voidaan lähitulevaisuudessa päivittää 3D-lasertutkiin, koska näiden hinnat ja suorituskyvyt alkavat lähestyä riittävää tasoa. 3D-lasertutkilla ympäristöstä saadaan kerättyä informaatiota paljon enemmän ja nopeammin. Lisäksi 3D-lasertutkien avulla voi olla mahdollista säästää muussa anturoinnissa. Esimerkiksi keskinivelen asentokulman voisi määrittää ajoneuvoon osuvista lasersäteistä. Lisäksi voitaisiin säästää esimerkiksi gyroskoopissa, jos sen ei enää tarvitse olla yhtä tarkka.

3.2 Käytettävät algoritmit

Tämän työn päärunkona toimii Krüsin et. al DPC-menetelmä [21], jossa käytetään pelkästään pistepilviä liikesuunnitteluun, liikeradan optimointiin ja maaston arviointiin. Seuraavassa luvussa käsitellään tiivistetysti heidän järjestelmänsä pääkohdat. Koska tutkimusryhmä ei ole julkaissut avointa lähdekoodia, tässä työssä on replikoitu osia heidän liikesuunnittelun algoritmeistaan, joita on muokattu runko-ohjattaville ajoneuvoille sopiviksi MATLAB-ympäristössä. Näitä muokkauksia käsitellään seuraavan luvun lopussa.

Tässä työssä pistepilvet muodostetaan myöhemmin käsiteltävällä Nelsonin avoimen lähdekoodin SLAM-algoritmilla [34], jota hän kutsuu BLAM-algoritmiksi. Hänen algoritminsa ei kuitenkaan ole niin pitkälle kehitetty kuin DPC-menetelmässä käytetty SLAM-algoritmi. Tässä työssä ei ole vielä tehty kovin suuria muutoksia hänen algoritmiinsa.

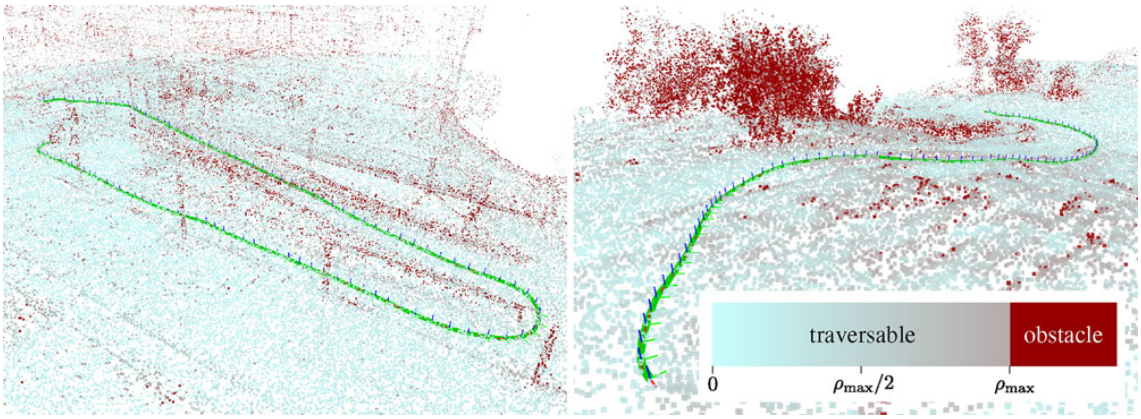
Viimeisenä algoritmina mainitaan hyvin lyhyesti Seegmillerin väitöskirjassaan [34] esittämä dynaamisten mallien formulointi pyörillä liikkuville roboteille. Tätä voidaan

mahdollisesti hyödyntää jatkotutkimuksessa, jos esimerkiksi halutaan simuloida nopeasti ajoneuvon dynamiikkaa joko tarkempaa liikesuunnittelua varten tai mahdollisesti turvallisuuden takia.

Vaihtoehtoisia menetelmiä ja algoritmeja on runsaasti. SLAM-algoritmeja on useita erilaisia, kuten edellisessä luvussa kerrottiin. Myös liikesuunnittelun algoritmeja on paljon. Navigoinnin voisi myös toteuttaa täysin eri lähtökohdista käyttämällä viime vuosien uusia innovatiivisia teknologioita.

3.3 Ajaminen pistepilvillä: liikesuunnittelu, liikeradan optimointi ja maaston arviointi yleispätevissä ei-tasomaisissa ympäristöissä

ETH Zürichin tutkimusryhmä, johon kuuluvat Krüsi, Furgale, Bosse ja Siegwart, esittää työssään [21] käytännöllisen lähestymistavan maarobottien globaalille liikkeen suunnittelulle ja maaston arvioinnille yleispätevissä 3D-ympäristöissä. Heidän työhönsä viitataan kirjainyhdistelmällä DPC (Driving on Point Clouds). Tässä luvussa käsitellään heidän artikkelinsa pääkohdat lyhyesti. Myöhemmin käsitellään, mitä muutoksia mahdollisesti vaadittaisiin, jotta samankaltaista järjestelmää voitaisiin hyödyntää myös kaivoskoneille. Kuvassa 23 on esitetty liikesuunnittelun ja maaston arvioinnin mallitulokset kaksikerroksisessa parkkihallissa ja pienessä mäessä.



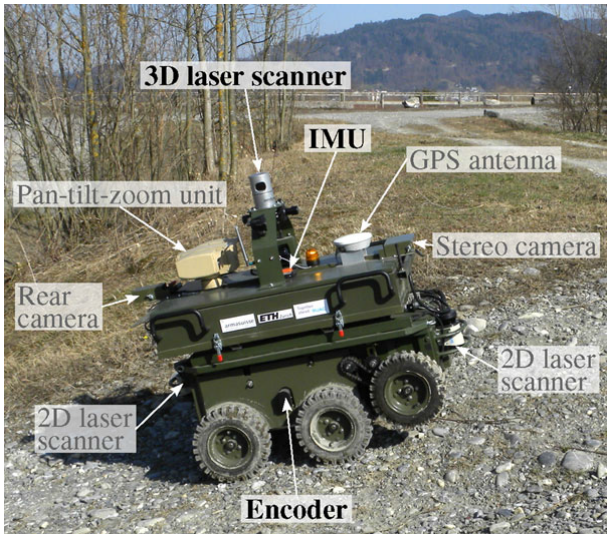
Kuva 23. *DPC-menetelmän liikesuunnittelun ja maaston arvioinnin demonstrointi kahdessa eri ympäristössä. Vasemmassa kuvassa on esitetty liikerata kaksikerroksisessa parkkihallissa ja oikeassa pienessä mäessä. Kuviin on laskettu jokaiselle pisteelle arvioitu maaston epätasaisuus visualisointia varten. Tummanpunaiset pisteet luokitellaan esteiksi. [21]*

DPC-menetelmä laskee optimoidun 6-ulotteisen liikeradan suoraan järjestämättömille pistepilville ilman mitään pinnanrekonstruktiota, diskretointia tai topologian erottamista. Liikerata noudattaa kaarevuus- ja jatkuvuusrajoituksia. Maaston geometriaa ja kulkukelpoisuutta arvioidaan tarvittaessa liikesuunnittelun aikana sovittamalla robotin kokoisia tasoja karttaan ja analysoimalla karttapisteiden paikallista jakautumaa. Liike-

suunnittelu koostuu kolmesta vaiheesta: ensiksi muodostetaan kahden RRT:n (Rapidly-exploring Random Tree) avulla alkuliikerata, jota parannetaan toisessa vaiheessa RRT*:een perustuvalla globaalilla optimoinnilla, ja lopuksi suoritetaan vielä hienojakoinen paikallinen optimointi. Näitä menetelmiä käytetään täysin autonomisessa navigaatiojärjestelmässä, joka perustuu 3D-lasertutkan ja ICP:n avulla suoritettavaan samanaikaiseen paikannukseen ja kartoitukseen (SLAM).

3.3.1 Järjestelmän yleiskuvaus

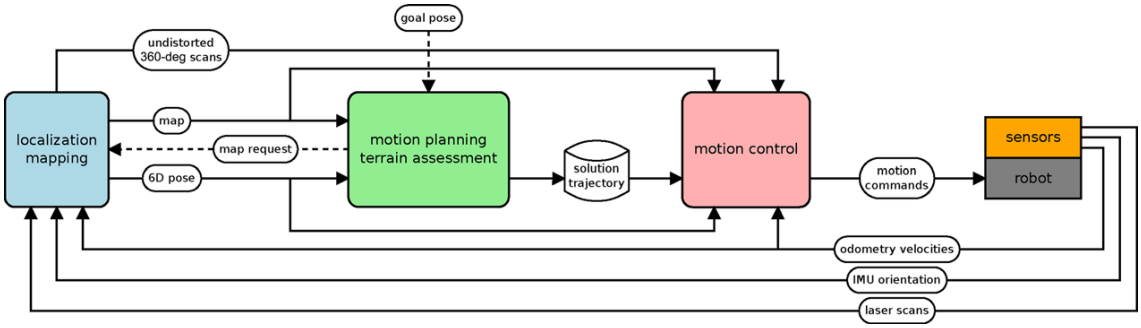
Kuvassa 24 on DPC-artikkelissa käyttämä robotti ARTOR ja siihen asennetut anturit. Näistä antureista käytetään kuitenkin vain 32-kanavaista 3D-lasertutkaa (Velodyne HDL-32E), inertiamittausyksikköä (Inertial Mesurement Unit, IMU, Xsens MTi) ja kahta asentoanturia pyörien nopeuksien mittaamiseen. Liukuohjattava ajoneuvolla on sähkömoottori, jonka avulla pyöritetään kolmea pyörää ajoneuvon kummallakin puolella. Ajoneuvo on suhteellisen pienikokoinen vain 1,3 metrin pituudellaan ja 0,7 metrin leveydellään. Ajoneuvon tietokoneen prosessori on kahdeksanytiminen Intel Core i7 ja käyttöjärjestelmä on Ubuntu Linux, jossa ajetaan ROSia (Robot Operating System). Anturidata on tarjolla ROS-viesteinä ja navigaatiojärjestelmän eri osat kommunikoivat keskenään ROS-viestien ja -palveluiden avulla. Robotin liikkeen ohjaukselle aiheuttaa ongelmia hidas ja epätarkka matalan tason moottorin ohjaus, joka toimii vain 10 Hz:n taajuudella ja sisältää tyypillisesti 0,4 sekunnin viiveen. Tämä aiheuttaa yhdessä laskennallisesti rajoitetun RRT*:n kanssa mutkia suorille avonaisille osuuksille.



Kuva 24. *ARTOR on liukuohjattava sähkömoottorilla varustettu robottiajoneuvo. DPC-tutkimusryhmän autonomisessa navigaatiojärjestelmässä käytetään ARTOR:n antureista vain 3D-lasertutkaa (Velodyne HDL-32E), inertiamittausyksikköä (IMU, Xsens MTi) ja kahta asentoanturia pyörien nopeuksien mittaamiseen. [21]*

Kuvassa 25 esitetään tutkimusryhmän yleiskuva autonomisen navigaatiojärjestelmän toiminnasta. Paikannuksen ja kartoituksen moduuli saa robotin antureilta pistepilvidataa sekä tiedot ajoneuvon asennosta ja pyörien nopeuksista. Moduuli tuottaa karttapohjaisen 6-ulotteisen paikannuksen sekä päivittää jatkuvasti sisäisiä karttojaan. Se myös toimittaa tarvittaessa pistepilven liikesuunnittelumoduulille. Tämä moduuli laskee liikeradan robotin sen hetkisestä paikasta määränpäähän käyttämällä liikesuunnittelun ja maaston arvioinnin menetelmiä, jotka kuvataan myöhemmissä luvuissa. Moduuli pyrkii suunnit-

telemaan liikerataa niin usein kuin on mahdollista. Liikkeen ohjauksen moduuli laskee sopivat liikekomennot, joilla robotti pyrkii seuraamaan laskettua liikerataa, ja varmistaa turvallisuuden pysäyttämällä ajoneuvon törmäyksen uhatessa. Moduuli toimii 10 Hz:n vakiotaajuudella.



Kuva 25. *DPC-tutkimusryhmän autonomisen navigaatiojärjestelmän toiminnan yleiskuva. Järjestelmä koostuu kolmesta osasta: paikannuksesta ja kartoituksesta, liikkeen suunnittelusta ja maaston arvioinnista sekä liikkeen ohjauksesta. Järjestelmä laskee antureiden avulla sopivat liikekomennot, joiden avulla robotti pystyy navigoimaan turvallisesti ja tehokkaasti käyttäjän määrittelemään päämäärään. [21]*

Ajoneuvon asennot esitetään 6-ulotteisessa avaruudessa jäykän kappaleen siirtojen ja rotaatioiden avulla käyttämällä 4×4 -muunnosmatriiseja, jotka kuuluvat erityiseen euklidiseen ryhmään $\mathrm{SE}(3)$. 6D-asento $\boldsymbol{T}_{AB}$ koordinaatistossa A voidaan ilmaista toisen koordinaatiston B avulla, jonka paikka ja orientaatio suhteessa A:han tiedetään, seuraavasti:

$$\boldsymbol{T}_{AB} = \begin{bmatrix} \boldsymbol{R}_{AB} & \boldsymbol{t}_{AB} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathrm{SE}(3). \tag{28}$$

Translaatio eli siirros ilmaistaan vektorina $\boldsymbol{t}_{AB} \in \mathbb{R}^3$, joka lähtee koordinaatiston A origosta ja päättyy B:n origoon. Rotaatiomatriisi $\boldsymbol{R}_{AB}$ kuuluu erityiseen ortogonaaliseen ryhmään $\mathrm{SO}(3)$ ja se ilmaistaan seuraavasti:

$$\boldsymbol{R}_{AB} = [\hat{\boldsymbol{x}}_{AB} \quad \hat{\boldsymbol{y}}_{AB} \quad \hat{\boldsymbol{z}}_{AB}] \in \mathrm{SO}(3). \tag{29}$$

$\boldsymbol{R}_{AB}$ kiertää vektorin koordinaatistosta B koordinaatistoon A, ja $\hat{\boldsymbol{x}}_{AB}$, $\hat{\boldsymbol{y}}_{AB}$ ja $\hat{\boldsymbol{z}}_{AB}$ ovat rotaatiomatriisin yksikkövektoreita, jotka määrittävät koordinaatiston B akseleita ilmaistuna koordinaatistossa A. Myöhemmässä vaiheessa käytetään ilmaisua T_{MR}, joka kuvaa ajoneuvon tai robotin koordinaatiston R sijaintia ja orientaatiota ilmaistuna karttakoordinaatistossa M.

Suunnittelukartta määritellään pisteiden joukkona, jossa on N_p pistettä, ja nämä pisteet esitetään muodossa $(\boldsymbol{p}, \boldsymbol{d}_{obs})$. Suorakulmaisen koordinaatiston sijainti ilmaistaan vektorilla $\boldsymbol{p} \in \mathbb{R}^3$ ja havainnointivektori $\boldsymbol{d}_{obs} \in \mathbb{R}^3$ ilmaisee, mistä suunnasta anturi on havainnut pisteen. Suunnittelukarttojen luokkaa merkitään notaatiolla $\mathfrak{M}_{pl}$ ja oletetaan li-

säksi, että pisteet voidaan ilmaista painovoiman mukaisessa koordinaatistossa ja että niiden tarkkuus sekä tiheys ovat riittävällä tasolla.

Liikesuunnitteluongelma määritellään seuraavasti: liikerata T, joka koostuu sarjasta robotin 6D-asentoja, lasketaan pistepilvikartassa $\mathcal{M} \in \mathfrak{M}_{pl}$ määritetyn lähtöasennon T_{MS} ja loppuasennon T_{MG} perusteella. Liikeradan tulee täyttää seuraavat rajoitukset:

1. Jokaisessa liikeradan asennossa ajoneuvolla täytyy olla maakosketus.
2. Jokaisen asennon täytyy olla staattisesti kulkukelpoinen, eli maaston epätasaisuus ρ ei saa olla liian suuri ajoneuvon kokoisella maaston alueella eivätkä ajoneuvon kaltevuuskulma ψ ja nousukulma θ saa ylittää rajoja:

$$\rho \leq \rho_{max} \in \mathbb{R}_{>0} \text{ sekä} \tag{30}$$

$$|\psi| \leq \psi_{max} \in \mathbb{R}_{>0} \text{ ja} \tag{31}$$

$$\mathbb{R}_{<0} \ni \theta_{min} \leq \theta \leq \theta_{max} \in \mathbb{R}_{>0}. \tag{32}$$

3. Robotin pitää pystyä seuraamaan liikerataa ilman sivuttaista liikettä.
4. Jokaisessa liikeradan pisteessä kaarevuuden täytyy pysyä alle ajoneuvolle ominaisen maksimin.
5. Liikeradan täytyy olla seurattavissa jatkuvalla nopeudella, koska ajoneuvo ei pysty muuttamaan ohjaustaan tai rengasnopeuksiaan äkkinäisesti.

Maaston arviointi on turvallisen ja optimaalisen liikesuunnittelun perusedellytys. Koska maa-ajoneuvot ovat rajoittuneet kulkemaan maaston pinnalla, ajoneuvon asennon kuudesta vapausasteesta kolme määräytyvät maaston paikallisen geometrian mukaan.

Maaston arvioinnin ongelma määritellään seuraavasti: annetulle 6D-asennolle ja tietylle ajoneuvon mallille etsitään lähin 6D-asento, joka sijaitsee maaston pinnalla, ja lasketaan maaston staattinen kulkukelpoisuuden arvo tässä asennossa. Menetelmä perustuu karttapisteiden geometrisen jakauman tilastolliseen laskemiseen paikallisessa ympäristössä.

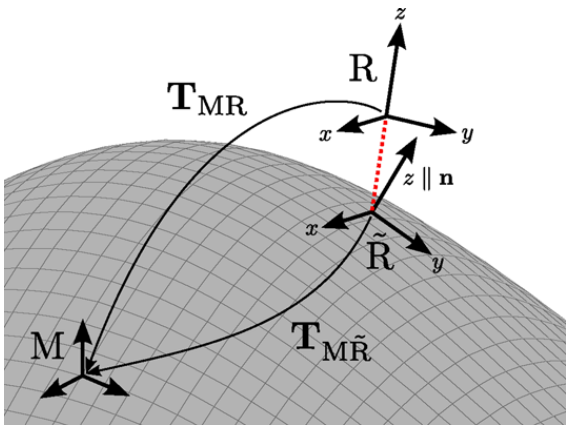
Maaston arvioinnin funktio f_{ta} jaetaan kahteen eri osaan:

$$\begin{matrix} f_{ta}^{p}\colon (\mathcal{M}, \boldsymbol{T}_{MR}) \rightarrow \boldsymbol{T}_{M\tilde{R}} \\ f_{ta}^{\tau}\colon (\mathcal{M}, \boldsymbol{T}_{M\tilde{R}}) \rightarrow \tau \end{matrix}, \tag{33}$$

jotka ovat maaston geometrian f_{ta}^{p} ja kulkukelpoisuuden f_{ta}^{τ} laskenta. Funktio f_{ta}^{p} laskee maaston pinnalla sijaitsevan 6D-asennon ja funktio f_{ta}^{τ} laskee kulkukelpoisuuden arvon tässä asennossa tietylle ajoneuvolle.

Funktion f_{ta}^{p} toiminnan perusperiaate on esitetty kuvassa 26. Annetulle 6D-asennolle $\boldsymbol{T}_{MR} \in \mathrm{SE}(3)$ etsitään lähin maaston pinnalla sijaitseva asento $\boldsymbol{T}_{M\tilde{R}} \in \mathrm{SE}(3)$ seuraavasti:

1. Asennon $\tilde{R}$ origo on lähin maaston pinnalla sijaitseva piste asennon R origosta asennon R z-akselilla.
2. Asennon $\tilde{R}$ z-akseli saadaan pinnan normaalista.
3. Loppujen akseleiden suunta määräytyy siten, ettei asentojen R ja $\tilde{R}$ välillä ole suunnan muutosta.



Kuva 26. *Maaston arvioinnin geometrisessa osassa etsitään annetulle asennolle $\boldsymbol{T}_{MR}$ lähin tuntemattoman maaston pinnnalla ja annetun asennon z-akselilla sijaitseva asento $\boldsymbol{T}_{M\tilde{R}}$. Asennot esitetään muunnosmatriiseina asentojen koordinaatistoista karttakoordinaatistoon. [21]*

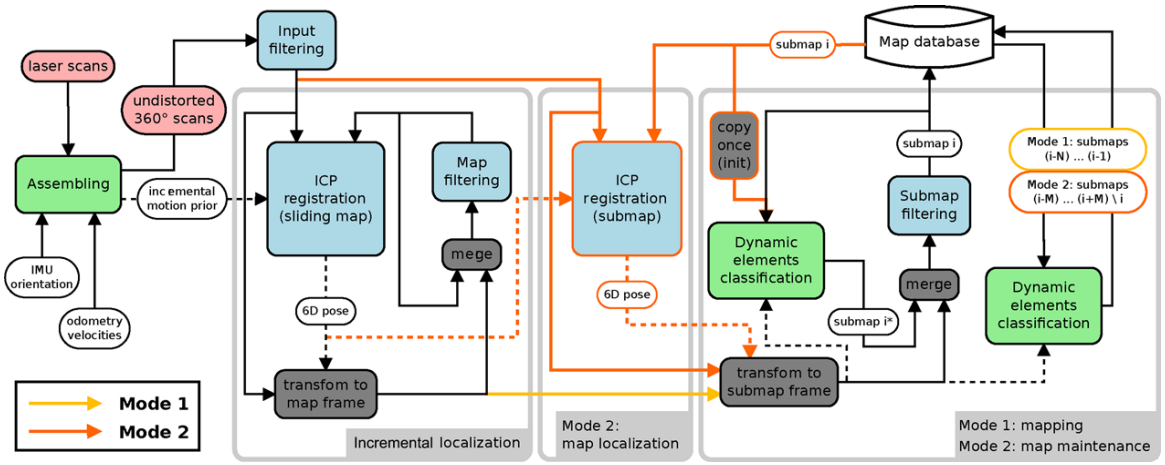
Funktion f_{ta}^{τ} on määritelty siten, että se huomioi maaston pinnan epätasaisuuden ja kaltevuuden. Tällöin se huomioi asetetut staattiset kulkukelpoisuusrajoitukset. Sen tuottama kulkukelpoisuuden estimaatti $\tau \in [0,1]$ tietylle asennolle kertoo, onko maaston kohta täysin kulkukelpoinen (1), kulkukelpoinen (> 0) vai kulkukelvoton (0).

Seuraavissa alaluvuissa tarkastellaan tarkemmin DPC-menetelmän eri moduulien toimintaa. Ensiksi tarkastellaan paikannusta ja kartoitusta. Seuraavissa alaluvuissa selitetään mahdollisimman yksityiskohtaisesti ja lyhyesti liikesuunnittelun ja maaston arvioinnin menetelmät, jotka noudattavat tässä alaluvussa esitettyjä rajoituksia. Lopuksi käsitellään vielä liikkeen ohjausta lyhyesti.

3.3.2 Paikannus ja kartoitus

DPC-tutkimusryhmä hyödyntää aikaisempia tutkimuksiaan ICP:hen perustuvassa paikannus- ja kartoitusjärjestelmässään. Järjestelmän yleiskuvaus löytyy kuvasta 27. Rekisteröintikirjasto (libpointmatcher) on saatavilla avoimena lähdekoodina. Järjestelmään on kuitenkin tehty kolme tärkeää lisäystä, jotka ovat lasertutkadatan vääristymien korjaus inertiamittausten perusteella, dynaamisten elementtien tunnistaminen ja staattisten osien erottaminen sekä jatkuva alikarttojen päivittäminen. Järjestelmässä ympäristö esitetään avaruudellisesti rajattuina alikarttoina, jotka muodostavat verkoston. Verkostossa alikarttoja yhdistävät reunat kuvaavat suhteellisia muunnoksia näiden välillä.

Järjestelmä voi toimia kahdessa eri tilassa, jotka ovat tutustuminen tuntemattomaan ympäristöön ja navigointi jo aiemmin kartoitetussa ympäristössä. Ensimmäisessä tilassa kartta rakennetaan käyttämällä vähittäistä paikantamista, jossa luodaan ja tallennetaan alikartat ajetulta reitiltä. Tunnetussa ympäristössä navigoidessa paikannetaan robotti käyttäen verkostoon tallennettuja alikarttoja.



Kuva 27. *ICP:hen perustuva paikannus- ja kartoitusjärjestelmä koostuu kahdesta toimintatilasta: tutustumisesta tuntemattomaan ympäristöön (tila 1) ja navigoinnista aiemmin kartoitetussa ympäristössä (tila 2). Lasertutkadatasta muodostetaan kokonaisia täyden kierroksen pistepilviä, jotka on korjattu mittauksen aikana tapahtuneen ajoneuvon liikkeen perusteella. Vähittäinen paikannus -moduuli ylläpitää rajoitettua karttaa, joka liikkuu robotin kanssa varmistaen jatkuvan toiminnan jopa dynaamisissa ympäristöissä. Tilassa 1 kartoitusmoduuli muodostaa verkoston avaruudellisesti rajoitetuista alikartoista, jotka on yhdistetty toisiinsa suhteellisilla muunnoksilla. Tilassa 2 paikannusmoduuli tuottaa korjatun pistepilven ja tietokannassa olevan lähimmän alikartan avulla paikannuksen. Samanaikaisesti ylläpitomoduuli päivittää alikarttoja. [21]*

Koska nykyisessä järjestelmässä käytetään pyörivää lasertutkaa, joudutaan sen tuottamaa dataa kokoamaan ja korjaamaan inertiamittausten mukaisesti, jotta saadaan vääristymätön täyden kierroksen pistepilvi. Ajoneuvon inertia- ja odometriamittausten avulla lasketaan anturin liike datapakettien välillä ja muunnetaan kaikki pisteet samaan koordinaatistoon jokaisella kierroksella. Tämä pistepilvi ja kokonaisen kierroksen yhdistetty liikkeen estimaatti, jota käytetään alkuarvauksena, syötetään ICP:hen perustuvaan vähittäiseen paikannukseen. Tämä moduuli samanaikaisesti muodostaa robotin mukana liikkuvan rajoitetun kokoisen liukuvan kartan, jossa se myös paikantaa robotin. Kartassa rajoitetaan pisteiden määrää ja tiheyttä.

Tutustumistilassa alikarttojen verkosto muodostetaan. Oikealla kuvassa 27 nähdään, että pistepilvet lisätään uusimpaan alikarttaan, kun ne on muunnettu alikartan koordinaatistoon käyttämällä vähittäisen paikantamisen tuottamaa asennon estimaattia. Nykyinen alikartta tallennetaan ja uusi alustetaan, kun kuljettu matka ylittää asetetun arvon tai kun nykyisen korjatun pistepilven ja alikartan päällekkäisyys alittaa asetetun arvon. Alikarttoja rajoitetaan myös tiheyden perusteella. Liikkuvien kohteiden aiheuttamien pisteiden takia täytyy jatkuvasti estimoida pisteiden dynaamisuutta. Estimaatin todennäköisyys kasvaa havaintojen määrän lisääntyessä. Tämän takia pisteiden dynaamisuuden todennäköisyyttä estimoidaan jatkuvasti myös aikaisempien alikarttojen osalta, jos ne ovat päällekkäin anturin näkökentän kanssa.

Kun navigoidaan aiemmin kartoitetussa ympäristössä, paikannusmoduuli lataa yhden alikartan kolmesta vaihtoehdosta alikarttaverkosta. Vaihtoehtoina ovat joko edellinen käytetty alikartta, sen edeltäjä tai sen seuraaja nykyisessä lineaarisessa verkostossa. Vaihtoehdoista valitaan se, mikä on kaikista lähimpänä robotin nykyistä paikkaa. Lisävaihtoehtona on uudelleenpaikannus, jos senhetkisen korjatun pistepilven ja valitun alikartan päällekkäisyys on liian pieni, jolloin etsitään tietyn säteen sisältä suurimman päällekkäisyyden antama alikartta. Tämän avulla robotin liike ei ole rajoitettu aiemmin kuljettuun reittiin.

Dynaamisissa ympäristöissä karttaan perustuva paikannus voi epäonnistua, minkä takia hyödynnetään vähittäistä paikannusta ja kartan ylläpitoa. Tällöin hyödynnetään liukuvaa karttaa kuten ympäristöön tutustumisvaiheessakin. Epäonnistumisen tapahtuessa käytetäänkin vähittäisen paikantamisen tuottamaa asennon estimaattia paikannuksen lopputuloksena. Normaaliin toimintaan yritetään palata heti, kun karttaan perustuva paikannus on jälleen toimiva. Karttojen ylläpito toimii paikannuksen kanssa rinnakkain ja päivittää jatkuvasti alikarttoja tietokannassa. Kuten kartoitusvaiheessakin aina, kun uusi alikartta on ladattu, karttadatat kopioidaan erilliseen prosessiin ja päivitetään. Samanaikaisesti pisteiden, jotka ovat sillä hetkellä käytettävän alikartan lähellä olevissa alikartoissa, dynaamisuuden todennäköisyyttä päivitetään ja välillä poistetaan erittäin dynaamiset pisteet. Alikarttoihin lisätään pisteitä, jos ympäristöön on tullut uusi kohde, ja niistä poistetaan pisteitä, jos kohde ei ole enää havaittavissa. Kun paikannus vaihtaa alikarttaa, päivitetty alikartta korvaa vanhan alikartan.

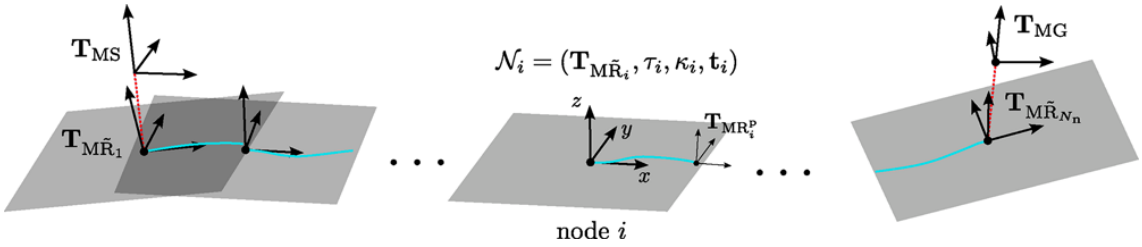
Paikannuksen ja kartoituksen tuottamia karttoja on tarkoitus hyödyntää liikesuunnittelussa. Suunnitteluun käytetyt kartat luodaan niitä tarvittaessa erikseen jokaiselle suunnittelupyynnölle, jolloin varmistetaan, että kartat sisältävät uusinta mahdollista tietoa. Ensimmäisessä vaiheessa alikarttojen verkostosta etsitään reitti robotin nykyisen paikan ja maalia lähimpänä olevan pisteen välille. Seuraavaksi kaikki reitillä olevat alikartat muunnetaan ensimmäisen alikartan painovoiman mukaiseen koordinaatistoon. Koska alikarttojen koko on paljon suurempi kuin keskimääräinen etäisyys niiden välillä, yhdistetyn kartan pistetiheys on paljon suurempi kuin yhden alikartan. Suunnittelukarttojen pistetiheys onkin noin 40 ja alikarttojen noin 10 pistettä kuutiometriä kohden. Tästä on paljon hyötyä maaston arvioinnissa. Suuri pisteiden tiheys lisää kuitenkin laskennallista taakkaa. Tämän vuoksi pisteitä on hyvä rajoittaa. Rajoittaminen tapahtuu poistamalla alikartoista kaikki ne pisteet, jotka ovat etäisyydeltään kauempana alikartan keskipisteestä kuin asetettu arvo. Tämä vähentää myös epätarkkuuksista johtuvia virheitä yhdistetyssä kartassa.

3.3.3 Liikeradan esittäminen liikesuunnittelussa

Liikesuunnittelu noudattaa DPC-artikkelissa määriteltyjä rajoituksia. Kuvassa 28 esitetään liikesuunnittelun tuottama liikerata, joka muodostuu sarjasta 6D-asentoja, jotka yhdistetään toisiinsa lyhyillä tasoliikeradoilla. Jokaista sarjan pistettä kutsutaan solmuksi

$\mathcal{N}_i$, joka koostuu maaston pinnalla sijaitsevan 6D-asennon $T_{M\tilde{R}}$ lisäksi kulkukelpoisuudesta τ_i, reitin kaarevuudesta κ_i ja kolmannen asteen kaarevuuspolynomista $\boldsymbol{t}_i$, joka määrittää tasoliikeradan seuraavaan solmuun. Solmujen välinen matka on yleensä lyhyempi kuin ajoneuvon pituus, jolloin liikerata pystyy noudattamaan paremmin ajoneuvon kinemaattisia rajoituksia. Merkitään liikesuunnittelijan liikeratojen luokkaa merkinnällä $\mathfrak{T}_{pl}$ ja määritellään se seuraavasti:

$$\mathfrak{T}_{pl} = \{(\mathcal{N}_i)_{i=1}^{N_n}\}, \qquad \mathcal{N}_i = (\boldsymbol{T}_{M\tilde{R}_i}, \tau_i, \kappa_i, \boldsymbol{t}_i). \tag{34}$$



Kuva 28. *Liikesuunnittelun tuottama ja ajoneuvolle sopiva liikerata ei-tasaisessa ympäristössä esitetään sarjana solmuja. Jokaiselle solmulle on määritelty 6D-asento $\boldsymbol{T}_{M\tilde{R}_i}$, kulkukelpoisuus τ_i, reitin kaarevuuden κ_i arvo solmussa ja kolmannen asteen kaarevuuspolynomi $\boldsymbol{t}_i$, jolla solmu yhdistetään seuraavaan approksimoimalla maastoa paikallisesti tasona. [21]*

Liikeradan $\mathrm{T} \in \mathfrak{T}_{pl}$ ensimmäisen ja viimeisen solmun asennot lasketaan projisoimalla käyttäjän määrittämät lähtö- ja loppuasennot maaston pinnalle seuraavasti:

$$\boldsymbol{T}_{M\tilde{R}_1} = f_{ta}^{p}(\mathcal{M}, \boldsymbol{T}_{MS}), \qquad \boldsymbol{T}_{M\tilde{R}_{N_n}} = f_{ta}^{p}(\mathcal{M}, \boldsymbol{T}_{MG}). \tag{35}$$

Solmujen väliset tasoliikeradat esitetään kolmannen asteen kaarevuuspolynomeina, jotka on käsitelty aikaisemmassa luvussa. Kaarevuuspolynomi esittää kaarevuutta etäisyyden funktiona ja kuvaa liikeratoja paikan, suunnan ja kaarevuuden mukaan jatkuvana seuraavasti:

$$\kappa(s) = \kappa_0 + as + bs^2 + cs^3, \qquad \kappa_0, a, b, c \in \mathbb{R}. \tag{36}$$

Liikeratasegmentti, jonka pituus on $s_f \in \mathbb{R}_{>0}$, määritellään parametrivektorina $\boldsymbol{t}$ seuraavasti:

$$\boldsymbol{t} = [\kappa_0 \quad a \quad b \quad c \quad s_f]^T. \tag{37}$$

Nagy ja Kelly [31] osoittavat, että kolmannen asteen kaarevuuspolynomit luontaisesti täyttävät ei-holonomiset liikkeen rajoitukset ja että niillä on riittävästi vapausasteita kahden tilan $\xi_{a,b} = [x \quad y \quad \phi \quad \kappa]_{a,b}^T$ yhdistämiseen avaruudessa, joka koostuu paikasta tasossa, suunnasta ja kaarevuudesta. Tasoliikerata $\boldsymbol{t}_i$ solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+1}$ välille lasketaan asentoon $\boldsymbol{T}_{M\tilde{R}_i}$ liitetyn koordinaatiston $\tilde{R}_i$ xy-tasossa, joka approksimoi paikallisesti maaston pintaa. Jos maasto ei ole tasaista, tasoliikeradan $\boldsymbol{t}_i$ lopussa olevan asennon $\boldsymbol{T}_{MR_i^p}$ ja seuraavan solmun $\mathcal{N}_{i+1}$ asennon $\boldsymbol{T}_{M\tilde{R}_{i+1}}$ välille syntyy epäjatkuvuus kohta. Oh-

jattavat osat, jotka ovat sijainti ja suunta tasossa, voidaan kuitenkin valita siten, että ne pysyvät jatkuvina tässä kohdassa. Jos alkutilan sijainti keskitetään solmuun ja valitaan sen suunnaksi nolla sekä sopiva kaarevuuden arvo, alkutilaksi saadaan koordinaatissa $\tilde{R}_i$ seuraava:

$$\xi_{i,a} = [0 \quad 0 \quad 0 \quad \kappa_i]^T. \tag{38}$$

Lopputila saadaan projisoimalla $\boldsymbol{T}_{M\tilde{R}_{i+1}}$ koordinaatiston $\tilde{R}_i$ xy-tasoon $\tilde{R}_i$:n z-akselia pitkin ja valitsemalla $\tilde{R}_i$:n ja projisoidun asennon x-akseleiden välinen kulma loppusuunnaksi $\Delta\phi$. Lisäksi asetetaan loppuasennon kaarevuudeksi seuraavan solmun kaarevuus. Tällöin saadaan seuraavaa:

$$\boldsymbol{T}_{\tilde{R}_i\tilde{R}_{i+1}} = \boldsymbol{T}_{M\tilde{R}_i}^{-1}\boldsymbol{T}_{M\tilde{R}_{i+1}} = \begin{bmatrix} r_{xx} & . & . & t_x \\ r_{xy} & . & . & t_y \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{39}$$

$$\xi_{i,b} = \begin{bmatrix} t_x & t_y & \text{atan2}(r_{xy}, r_{xx}) & \kappa_{i+1} \end{bmatrix}^T. \tag{40}$$

Jos halutaankin löytää $\boldsymbol{T}_{M\tilde{R}_{i+1}}$ tietyn tasolla olevan pisteen $\boldsymbol{T}_{MR_i^p}$ avulla, voidaan käyttää maaston arvioinnin funktiota f_{ta}^p. Näillä menetelmillä täytetään jatkuvuudelle asetetut ehdot.

Tutkittaessa muita rajoituksia yleispätevälle ei-tasomaiselle liikeradalle tehdään seuraava sileysoletus: on olemassa sellainen etäisyys $r_{s,max} > 0$ siten, että jollekin etäisyydelle r_s $\left(0 < r_s < r_{s,max}\right)$ on mahdollista löytää kulma $\gamma_{max}^n(r_s)$, joka on paljon pienempi kuin $\frac{\pi}{2}$ ja jota ajoneuvon z-akselin orientaation muutos maaston arvioinnin funktion f_{ta}^p avulla laskettuna ei ylitä missään pisteessä tämän etäisyyden r_s sisällä. Lisäksi oletetaan, että kulma $\gamma_{max}^n(r_s)$ pienenee etäisyyden r_s pienetessä. Funktio f_{ta}^p tasoittaa hieman suuriakin esteitä ja seiniä ajoneuvon kokoluokan mukaan, minkä takia edelliset oletukset pätevät. Kun valitaan liikeradan solmujen välinen maksimietäisyys d_{IN}^{max} pienemmäksi kuin $r_{s,max}$, voidaan varmistaa, ettei ajoneuvon z-akselin ja pinnan approksimoidun normaalin välinen kulma ylitä koskaan arvoa $\gamma_{max}^n(d_{IN}^{max})$.

Ajoneuvon toteuttaessa liikerataa T sen todellinen reitti voidaan nähdä liikeradan projektiona tasolle missä tahansa liikeradan pisteessä. Määritelmien perusteella sijainti ja suunta ovat jatkuvia. Kuitenkin tasoliikeratojen kaarevuus muuttuu projektiossa, minkä takia siirryttäessä solmukohdasta toiseen kaarevuus voi olla epäjatkuva. Solmujen välillä kaarevuus on jatkuva, mutta se saattaa ylittää raja-arvon. Voidaan osoittaa, että kaarevuus missä tahansa tasoliikeradan pisteessä kasvaa enintään kertoimella $1/\cos\gamma$ tai laskee kertoimella $\cos\gamma$, jos projisoitua tasoa on käännetty kulman γ verran. Tällöin todellisen kaarevuuden κ_{act} ja suunnitellun kaarevuuden κ_{plan} välinen suhde millä tahansa liikeradan pisteellä on jollakin seuraavalla välillä:

$$\frac{\kappa_{act}}{\kappa_{plan}} \in \left[\cos\gamma_{max}^{n}(d_{IN}^{max}), \frac{1}{\cos\gamma_{max}^{n}(d_{IN}^{max})}\right]. \quad (41)$$

Jos solmujen välinen etäisyys d_{IN}^{max} valitaan tarpeeksi pieneksi, kaarevuuden epäjatkuvuuden maksimiarvot pienenevät, jolloin liikerata suunnilleen noudattaa kaarevuuden jatkuvuusrajoitusta. Jotta kaarevuuden maksimiarvoa ei ylitetä, vähennetään suunnitteluvaiheessa sallittua kaarevuuden maksimiarvoa seuraavasti:

$$\kappa_{max} = \kappa_{max,act}\cos\gamma_{max}^{n}(d_{IN}^{max}). \quad (42)$$

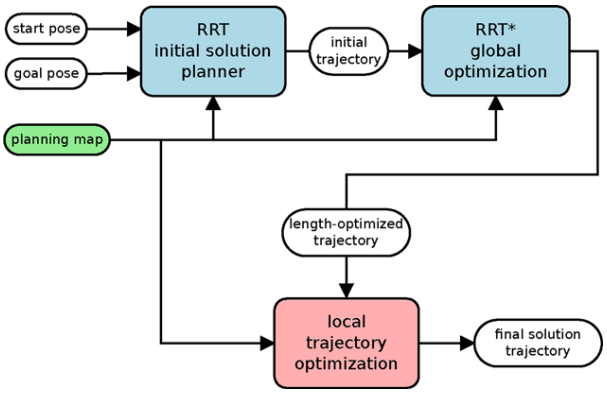
Rajoitetaan samoin myös suunnitteluvaiheen sallittuja rajoja ajoneuvon kallistuskulmalle ja nousukulmalle:

$$\begin{aligned}\psi_{max} &= \psi_{max,act} - \gamma_{max}^{n}(d_{IN}^{max})\\ \theta_{max} &= \theta_{max,act} + \gamma_{max}^{n}(d_{IN}^{max})\\ \theta_{min} &= \theta_{min,act} - \gamma_{max}^{n}(d_{IN}^{max}).\end{aligned} \quad (43)$$

3.3.4 Liikesuunnittelun yleiskuvaus

DPC-menetelmän liikesuunnittelun menetelmän kolme päävaihetta on esitetty kuvassa 29. Ensimmäiseksi käytetään kahta RRT:tä alkuliikeradan muodostamiseksi, sitten liikerata optimoidaan RRT*:n avulla ja lopullinen liikerata muodostetaan käyttäen hienojakoista paikallista optimointia. Liikeradat noudattavat edellisessä luvussa esitettyjä rajoituksia ja muotoilua.

Alkuliikerata muodostetaan laajentamalla kahta RRT:tä, joista toinen lähtee lähtöasennosta ja toinen loppuasennosta. Liikerata muodostuu ensimmäisestä onnistuneesta ratkaisusta, joka on ajoneuvon rajoitusten kanssa yhteensopiva eikä sisällä törmäyksiä ympäristön kanssa. Tämä liikerata voi kuitenkin sisältää huomattavia kiertoja, minkä takia toisessa vaiheessa käytetään RRT*:een perustuvaa algoritmia optimoimaan liikeradan pituutta. RRT* tuottaa asymptoottisesti optimaalisia liikeratoja, mutta sen ongelma on pitkä laskenta-aika. Sen takia RRT*:n toimintaa rajoitetaan eikä sitä käytetä heti ensimmäisessä vaiheessa. Tämän takia saatu tulos on



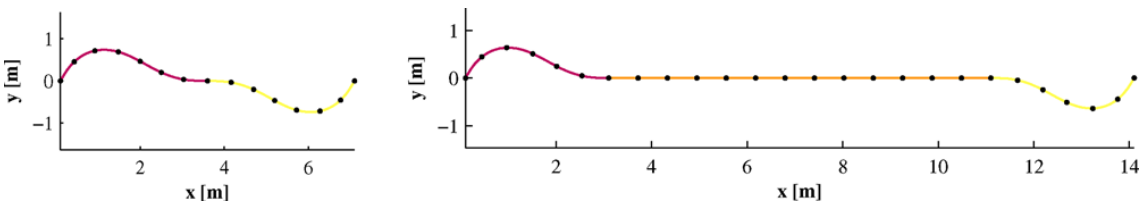
Kuva 29. *Liikesuunnittelussa liikeradan laskeminen suoritetaan kolmessa vaiheessa. Ensiksi alkuliikerata lasketaan kahden RRT:n avulla, toiseksi liikerata optimoidaan RRT*:n avulla ja lopuksi liikerata optimoidaan paikallisesti käyttäjän antaman kustannusfunktion mukaisesti. [21]*

kompromissi laskentanopeuden ja optimaalisuuden välillä. Tähän mennessä laskettu reitti saattaa olla lähellä lyhintä reittiä, mutta se luultavasti sisältää huonoja valintoja reitin kaarevuuden ja maaston kulkukelpoisuuden kannalta. Kaarevuuden ja maaston kulkukelpoisuuden tarkastelu voitaisiin teoriassa yhdistää myös RRT*:n kanssa, jolloin saataisiin erittäin tarkkoja tuloksia, mutta tällöin laskentanopeus hidastuisi. Tämän takia käytetään hienojakoista paikallista käyttäjän kustannusfunktioon perustuvaa optimointia. Siihen kuuluvat reitin kaarevuus ja pituus sekä maaston kulkukelpoisuus. Reittiä muunnellaan hieman ja etsitään paikallinen minimi.

3.3.5 Lyhyiden matkojen liikeratojen muodostaminen

Lyhyille matkoille on mahdollista käyttää yksinkertaista ja nopeaa liikeradan muodostusta. Menetelmässä oletetaan aluksi, että alku- ja loppuasennon välissä oleva maasto on kulkukelpoista ja 2,5-ulotteista eli se voidaan ilmaista kuten DEM-kartta (Digital Elevation Map), jossa ei ole useita kerroksia. Toisessa vaiheessa tarkastetaan liikeradan toteutuskelpoisuus, jonka perusteella liikerata joko hyväksytään tai hylätään. Menetelmää käytetään kahdessa eri kohteessa. Ensimmäinen käyttökohde on yhdistää kaksi RRT:tä toisiinsa alkuliikerataa muodostettaessa. Toinen käyttökohde on RRT*:een perustuvassa optimoinnissa, kun yhdistetään puun pisteitä uusiin näyteasentoihin.

Ensiksi algoritmissa projisoidaan loppuasento alkuasennon tasolle. Sitten riippuen asentojen välisestä etäisyydestä käytetään joko yhtä, kahta tai kolmea kolmannen asteen kaarevuuspolynomia yhdistämään asennot tasossa. Suuremmilla etäisyyksillä kannattaa käyttää useampaa polynomia, jotta tuotettu liikerata ei olisi tarpeettoman pitkä. Tasoliikerata jaetaan tasavälein asentoihin, jotka projisoidaan maaston pinnalle. Asentojen välinen etäisyys pidetään yhtä suurena tai pienempänä kuin valittu nimellinen solmujen välinen etäisyys d_{IN}^{nom}. Jos kaikki solmut ovat kulkukelpoisessa maastossa, tarkistetaan, että loppupiste on oikeassa paikassa. Jos ympäristö sisältää useita kerroksia, tämä piste voi olla väärässä kerroksessa. Seuraavaksi lasketaan solmujen välisten liikeratojen kolmannen asteen kaarevuuspolynomit. Lopuksi tarkastetaan, ettei reitin kaarevuus ylitä kynnysarvoa. Kuvassa 30 on esimerkki lyhyen matkan liikeradan muodostuksesta yhdellä ja kahdella välipisteellä.



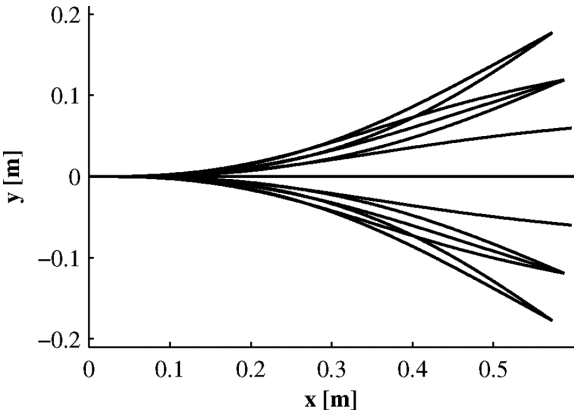
Kuva 30. *Lyhyillä matkoilla maastoon sopivat liikeradat suunnitellaan ensiksi muodostamalla tasoliikerata yhdistämällä kaksi asentoa, joiden sijainti ja suunta on määritelty. Liikerata saadaan laskettua jakamalla tasoliikeradat tasavälein asentoihin ja projisoimalla ne maaston pinnalle. Lyhyet matkat voidaan esittää yhdellä kolmannen asteen kaarevuuspolynomilla, ja pidemmillä matkoilla voi olla yksi tai kaksi välipistettä matkan lyhentämisen takia. [21]*

3.3.6 Alkuliikeradan muodostaminen RRT:llä

Alkuliikerata muodostetaan yhdistämällä alku- ja loppuasento toisiinsa kahden RRT:n avulla. Tätä ei ole vielä optimoitu millään kriteereillä, mutta se ei sisällä törmäyksiä ja noudattaa ajoneuvon ja maaston rajoituksia. Tarkoituksena on muodostaa tämä liikerata mahdollisimman nopeasti ja käyttää sitä pohjana optimoiduille ratkaisuille. Sekä alkuasennosta että loppuasennosta lähtevää RRT:tä kasvatetaan samanaikaisesti. Kun eri puiden pisteiden etäisyys alittaa etäisyyden kynnysarvon, puut yritetään yhdistää näiden pisteiden kautta toisiinsa. Seuraavaksi käydään läpi, kuinka näytteenottaminen, puun laajentaminen ja puiden yhdistäminen tapahtuvat.

Näytteenotto voisi tapahtua täysin sattumanvaraisesti koko tila-avaruudesta tai painottamalla puun laajentamista kohti loppua tai toista puuta. Jälkimmäinen vaihtoehto saattaa olla hidas, jos alku- ja loppuasento ovat avaruudellisesti lähellä toisiaan mutta niiden välillä on seinä tai ne ovat eri kerroksissa, jolloin ne voidaan yhdistää toisiinsa vain laajoilla kierroilla. Näytteenottaminen tapahtuu kuitenkin satunnaisesti mutta käyttäen vain suunnittelukartan pisteitä. Jokaista kartan pistettä voidaan käyttää liikeradan suunnitteluun eikä puun laajentaminen ole mahdollista kartan ulkopuolelle. Lisäksi suunnittelukartat voidaan muodostaa myös siten, että ne sisältävät vain alueen alku- ja loppuasennon väliltä, jolloin laajentaminen painottuu luonnollisesti kohti toista puuta.

Kun kartasta on otettu jokin näytepiste $\boldsymbol{p}_{samp}$, etsitään puusta lähin sitä oleva piste v_{near}. Seuraavaksi yritetään laajentaa puuta tästä lähimmästä pisteestä kohti näytepistettä mutta laskennallisesti työläiden toimenpiteiden, kuten liikeratojen muodostamisen tai simuloinnin, välttämiseksi käytetään ennalta laskettuja tasoliikeratoja, joista on esimerkki kuvassa 31. Aluksi kaikki liikeradat liitetään puun pisteeseen v_{near}. Näistä valitaan kuitenkin se, joka suuntautuu sopivimmin puun pisteen v_{near} ja näytepisteen muodostaman suoran kanssa. Koska laajennosten pituudet on valittu tarpeeksi lyhyiksi verrattuna ajoneuvon kokoon, voidaan uusi mahdollinen puun piste laskea käyttämällä maaston arvioinnin funktioita. Jos piste on kulkukelpoinen, lisätään se puuhun ja merkitään valittu tasoliikerata käyttökelvottomaksi pisteestä v_{near} tulevia puun laajennoksia varten. Kun kaikki tasoliikeradat on käytetty puun jostakin pisteestä v_i, piste poistetaan kokonaan laajentamisvaiheesta. Valmiiksi lasketut tasoliikeradat muo-



Kuva 31. *RRT:n laajentamisessa voidaan käyttää kuvan mukaisia tasoliikeratoja. Kuvan liikeratojen kaarevuus alussa ja lopussa on määritelty nollaksi, jotta yhdistetty liikerata olisi saumaton. Suunnan maksimaalinen muutos on riippuvainen kaarevuuden ylärajasta. Liikeradat kannattaa valita siten, että loppusuunnat jakautuvat sopivasti koko alueelle. [21]*

dostuvat kolmannen asteen kaarevuuspolynomeista. Jotta kaarevuus olisi jatkuva koko puussa, kaikkien tasoliikeratojen kaarevuus alussa ja lopussa on määritelty nollaksi. Tasoliikeradat määritellään diskreeteillä loppupisteillä, jotka sijaitsevat tietyn kokoisen ympyrän kehällä. Lisäksi loppusuunnat ovat diskreettejä. Tasoliikeratojen kaarevuus ei saa ylittää raja-arvoaan, ja niistä valitaan ne, joiden kaarevuuden maksimiarvo on mahdollisimman pieni. Tasoliikeradat ovat siis kaarevuudeltaan rajoitettuja, ja jokainen niistä tuottaa erilaisen suunnan muutoksen.

Puiden yhdistämistä varten ylläpidetään etäisyysmatriisia alku- ja loppupuun pisteiden välillä. Näitä puita pyritään yhdistämään jokaisella iterointikierroksella, jos joidenkin eri puiden pisteiden välinen etäisyys on vähemmän kuin määrätty kynnysarvo. Yhdistäminen tapahtuu aikaisemmin tässä luvussa esitetyllä lyhyen matkan liikeradan muodostamismenetelmällä. Yhdistäminen voi epäonnistua, jos maasto ei ole kulkukelpoista tai kaarevuuden kynnysarvo ylittyy. Tällöin pistepari merkitään toteuttamiskelvottomaksi tulevaisuuden yhdistämisyrityksiä varten. Lopullinen alkuliikerata saadaan yhdistämällä puiden liikeradat.

3.3.7 Liikeradan globaali optimointi RRT*:llä

Alkuliikerata saattaa vielä sisältää merkittäviä kiertoja. Liikeradan pituutta pyritään optimoimaan RRT*:een perustuvalla optimoinnilla. Myös muita optimointikriteerejä voisi käyttää. Alkuliikeradan oletetaan olevan kohtuullisen lähellä optimaalista reittiä. Jokaisella iterointikierroksella puuta pyritään laajentamaan satunnaisen pisteen suuntaan. Tämän takia satunnaisten pisteiden valinta rajoitetaan alkuliikeradan ympäristöön. Aluksi valitaan jokin piste v_{rand} puusta, jonka pallomaisesta ympäristöstä valitaan jokin satunnainen piste $\boldsymbol{p}_{samp}$. Ympäristön säde on verrannollinen lähtö- ja loppupisteen väliseen etäisyyteen.

Puuta laajennetaan seuraavasti: Ensiksi etsitään pisteen $\boldsymbol{p}_{samp}$ lähin naapuri v_{near} puusta. Näiden välinen suora projisoidaan pisteen v_{near} tasolle. Alkuasento pisteelle v_{new} sijoitetaan suoralle määrätyn etäisyyden päähän pisteestä v_{near}, minkä jälkeen pisteen todellinen asento lasketaan maaston arviointi funktioilla. Jos piste sijaitsee kulkukelpoisessa maastossa, etsitään kaikki sitä tietyn etäisyyden sisällä olevat puun pisteet, jotka muodostava joukon V_{near}. Jokaisesta pisteestä joukossa V_{near} yritetään laskea liikerata pisteeseen v_{new} aikaisemmin tässä luvussa esitetyllä lyhyen matkan liikeradan muodostamismenetelmällä. Kaikista mahdollisista liikeradoista valitaan se, jonka kokonaiskustannus on pienin pisteessä v_{new}. Tämän jälkeen lasketaankin pisteestä v_{new} liikeradat kaikkiin joukon V_{near} pisteisiin. Jos jonkun reitin kustannus on pienempi pisteen v_{new} kautta joukon pisteeseen $v_{near,i}$ kuin nykyinen reitti tähän pisteeseen, vaihdetaan tämän pisteen edeltäjä pisteeseen v_{new} ja liikerata lisätään puuhun. Koko liikerata kehittyy vähitellen aina, kun alkuperäisen puun piste voidaan reitittää uudelleen. Kuva 12 havainnollistaa tätä, vaikka siinä käytetään muuttujilla eri nimiä.

Menetelmän päättämiskriteerit muodostavat kompromissin optimaalisuuden ja laskenta-ajan kanssa. Menetelmän päättämisessä käytetään liikkuvaa keskiarvoa N_k^{avg} joukon V_{near} pisteiden määrästä N_k jokaisella iterointikierroksella seuraavasti:

$$N_k^{avg} = \alpha N_k + (1-\alpha)N_{k-1}^{avg}, \qquad 0 < \alpha \ll 1. \tag{44}$$

Tämän avulla voidaan optimointi pysäyttää, kun merkittäviä parannuksia ei ole enää mahdollisesti saavutettavissa. Kun liukuva keskiarvo ylittää valitun kynnyksen, tiedetään, että alkuliikeradan ympäristö on tutkittu tarpeeksi tiheästi. Tämä menetelmä sopii eripituisiin liikeratoihin ja erilaisiin maastoihin. Lopullinen liikerata saadaan jäljittämällä reitti puun lopusta alkuun.

3.3.8 Liikeradan paikallinen optimointi

Tähän mennessä saatu liikerata saattaa sisältää turhan jyrkkiä käännöksiä tai liian hankalia maaston osia. Paikallisessa liikeradan optimoinnissa käytetään asetettua kustannusfunktiota reitin hienosäätöön. Kustannus on määritelty rajoitetuksi vain toteuttamiskelpoisille liikeradoille, jotka noudattavat kulkukelpoisuus- ja kaarevuusrajoituksia. Kelpaamattomille liikeradoille kustannus on ääretön.

Itse kustannusfunktio on summa koko reitin liikeratasegmenttien painotetuista summista, jotka koostuvat reitin pituudesta, kaarevuudesta ja käänteisestä kulkukelpoisuudesta. Laskemisen tehostamiseksi käytetään kuitenkin muutamia yksinkertaistuksia:

1. Solmujen välinen etäisyys pidetään lähellä valittua nimellistä arvoa $d_{IN}^{nom} \in \left[d_{IN}^{min}, d_{IN}^{max}\right]$.
2. Liikeratasegmentin pituus on lähellä solmujen välistä todellista etäisyyttä.
3. Kulkukelpoisuus liikeratasegmentillä on lähellä sen yhdistämien solmujen kulkukelpoisuutta.
4. Kaarevuuden itseisarvon maksimeja halutaan karsia tai pienentää.

Kustannusfunktio liikeradalle $\mathrm{T} \in \mathfrak{T}_{pl}$ on muotoa

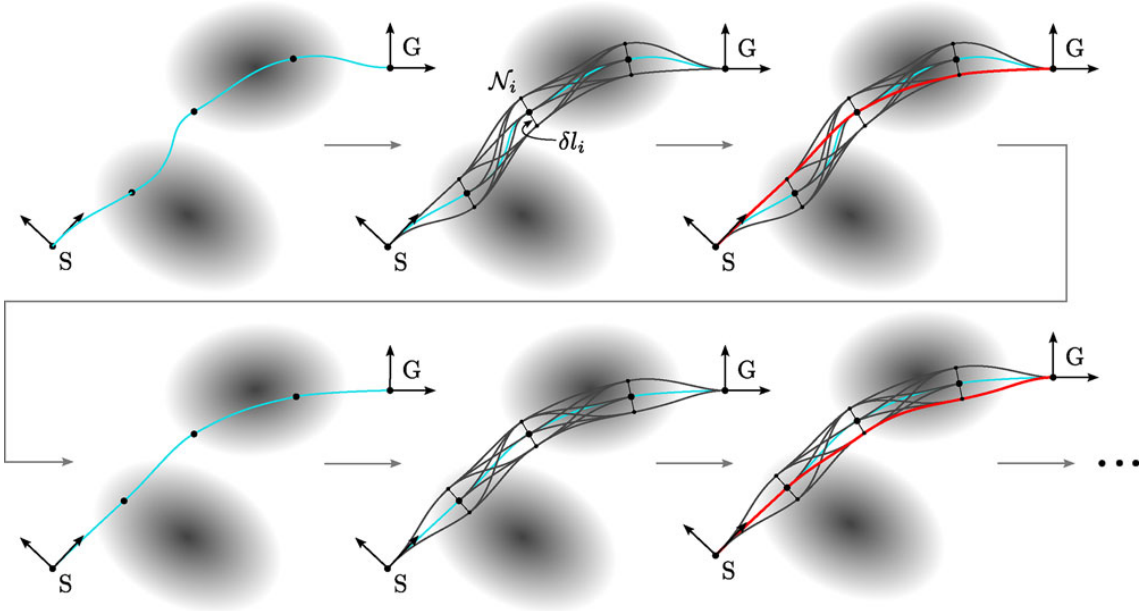
$$f_{cost}(\mathrm{T}) = \sum_{i=1}^{N_n-1} c_i, \tag{45}$$

missä N_n on solmujen lukumäärä ja c_i on solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+1}$ välisen segmentin kustannus. Kustannus c_i lasketaan seuraavasti:

$$c_i = \begin{cases} \infty, & \text{jos } \kappa_{\max}(\boldsymbol{t}_i) > \kappa_{\max} \text{ tai } \tau_{i+1} = 0 \\ \omega_{leng}\dfrac{d_i^{i+1} - d_{IN}^{min}}{d_{IN}^{max} - d_{IN}^{min}} + \omega_{curv}\dfrac{\kappa_{\max}(\boldsymbol{t}_i)}{\kappa_{\max}} + \omega_{trav}(1-\tau_{i+1}), & \text{muuten.} \end{cases} \tag{46}$$

Kahden solmun välinen c_i lasketaan painotettuna summana käyttämällä solmujen välistä etäisyyttä d_i^{i+1} (normalisoidaan solmujen välisten minimi- ja maksimietäisyyksien avulla), suurinta kaarevuuden itseisarvoa $\kappa_{\max}(\boldsymbol{t}_i)$ tasoliikeradalla ja käänteistä kulkukelpoisuutta $1-\tau_{i+1}$. Jokainen näistä arvoista on normalisoitu välille [0,1]. Kustannus on ääretön, jos kaarevuus $\kappa_{\max}(\boldsymbol{t}_i)$ ylittää maksimiarvon $\kappa_{\max}$ tai reitti ei ole kulkukelpoinen, jolloin τ_{i+1} on 0. Optimoinnin tulos on kompromissi liikeradan pituuden, kaarevuuden ja riskien osalta. Tulosta voidaan ohjata haluttuun suuntaan muuttamalla painokertoimia ω.

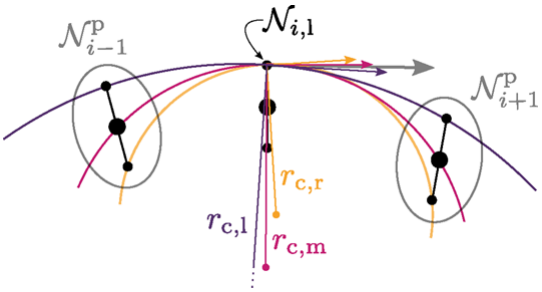
Kuvassa 32 esitetään paikallisen optimoinnin pääperiaate. Liikerataa muokataan pienin askelin iteratiivisesti kustannusfunktion avulla, kunnes paikallinen minimi on saavutettu. Jokaisella iterointikierroksella jokaisen solmun oikealle ja vasemmalle puolelle kulkusuuntaan nähden asetetaan pienen sivuttaisen matkan δl_i päähän alisolmu. Näin ollen jokainen solmu koostuu kolmesta alisolmusta mukaan lukien alkuperäinen solmu. Kun nämä yhdistetään seuraavan solmun alisolmuihin, muodostuu yhdeksän liikerataa, jotka esitetään kolmannen asteen kaarevuuspolynomien avulla. Kokonaisuudessa muodostuu 3^{N_n-2} vähän erilaista liikerataa alusta loppuun. Seuraavaa iterointikierrosta varten tästä liikeratojen muodostamasta verkosta etsitään pienimmän kustannuksen tuottava



Kuva 32. *Paikallisessa liikeradan optimoinnissa jokaisella iterointikierroksella, joita on tässä kuvassa kaksi, muodostetaan verkko vaihtoehtoisista liikeradoista lähdöstä loppuun. Tämä tehdään siten, että lisätään jokaisen solmun molemmin puolin pienen matkan päähän liikeradasta alisolmu. Tällöin saadaan 3^{N_n-2} vähän erilaista liikerataa, joista etsitään pienimmän kustannuksen tuottava liikerata. Kuva on esitetty selvyyden vuoksi vain 2D-tasossa. Harmaat alueet esittävät kulkukelpoisuudeltaan huonompia alueita. [21]*

liikerata, joka noudattaa kulkukelpoisuus- ja kaarevuusrajoituksia. Iterointi lopetetaan,

kun muut vaihtoehdot eivät enää vähennä kustannusta, jolloin paikallinen minimi on saavutettu.



Kuva 33. *Suunnan ja kaarevuuden laskeminen alisolmulle tapahtuu projisoimalla vierekkäiset alisolmut samalle tasolle ja laskemalla keskiarvot muodostuneiden ympyröiden tangenteista sekä kaarevuuksista. Kuvassa käytetään esimerkkinä vasemmanpuoleista alisolmua* $\boldsymbol{\mathcal{N}}_{i,l}$*. [21]*

Alisolmut sijoitetaan oikealle ja vasemmalle kulkusuuntaan nähden pienen matkan päähän liikeradan solmuista. Sivuttainen matka δl_i riippuu solmun kaarevuudesta ja asetetuista raja-arvoista. Jos solmun kaarevuus on suuri, käytetään pienempää sivuttaista matkaa, jolloin kyseisten liikeratasegmenttien maksimaalinen kaarevuus pienentyy. Ensiksi esimerkiksi kulkusuuntaan nähden vasemmanpuoleisen alisolmun $\mathcal{N}_{i,l}$ sijainti alustetaan solmun asennon $T_{M\tilde{R}_i}$ origosta y-akselille pienen matkan δl päähän. Seuraavaksi käytetään maaston arvioinnin menetelmiä maaston mukaisen asennon selvittämiseksi.

Koska tätä alisolmua edeltävä ja seuraava alisolmu voi olla mikä tahansa solmujen $\mathcal{N}_{i-1}$ ja $\mathcal{N}_{i+1}$ alisolmuista, valitaan suunnaksi ja kaarevuudeksi jokaiselle tämän alisolmun kautta kulkevalle liikeradalle sopiva kompromissi. Kompromissin etsintää havainnollistetaan vasemmanpuoleiselle alisolmulle kuvassa 33. Viereiset alisolmut projisoidaan kaikki samalle alisolmun tasolle ja näiden kautta lasketaan kolme ympyrää. Alisolmun suunnaksi määritetään tangenttivektoreiden keskiarvo. Kaarevuus määritellään ympyröiden säteiden avulla määritettyjen etumerkillisten kaarevuuksien keskiarvona. Etumerkki määritetään ympyrän keskipisteen sijainnin perusteella siten, että kulkusuuntaan nähden oikealla olevat ovat negatiivisia ja vasemmalla positiivisia. Lopuksi lasketaan vielä alisolmun kulkukelpoisuus.

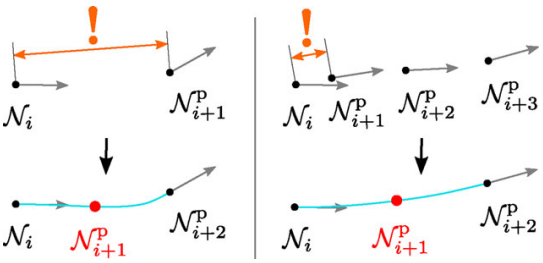
Liikeradan optimoinnin alussa kaikki solmujen väliset etäisyydet ovat suunnilleen nimellisiä johtuen menetelmästä. Optimoinnin aikana solmujen väliset etäisyydet muuttuvat liikerataan tehtävien muutosten takia. Jos etäisyydet ovat liian lyhyitä, tehdään mahdollisesti paljon turhaa laskentaa. Etäisyyksien ollessa liian pitkiä ajoneuvon rajoitusten noudattaminen voi estyä. Tämän vuoksi valitaan minimi- ja maksimietäisyydet (d_{IN}^{min} ja d_{IN}^{max}) solmujen välille. Jokaisella iterointikierroksella aina ennen kuin lisätään alisolmuja, tarkastetaan solmujen väliset etäisyydet. Jos raja-arvot ylitetään tai alitetaan, solmuja lisätään tai poistetaan, kuten kuvassa 34 havainnollistetaan. Solmua lisätessä solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+1}$ väliin projisoidaan jälkimmäinen ensimmäisen tasoon. Näiden välille muodostetun tasoliikeradan puoleenväliin lisätään uusi solmu ja lasketaan maaston arvioinnin menetelmillä maaston mukainen asento. Suunta, kaarevuus ja kulkukelpoisuus lasketaan kuten aikaisemmin alisolmuille mutta käyttäen vain yhtä ympyrää kolmen sijasta. Solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+1}$ ollessa liian lähellä toisiaan poistetaan solmun $\mathcal{N}_{i+1}$ myös

seuraava solmu $\mathcal{N}_{i+2}$. Tämän jälkeen lisätään solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+3}$ väliin uusi solmu samalla tavalla kuin liian pitkän välin tapauksessa.

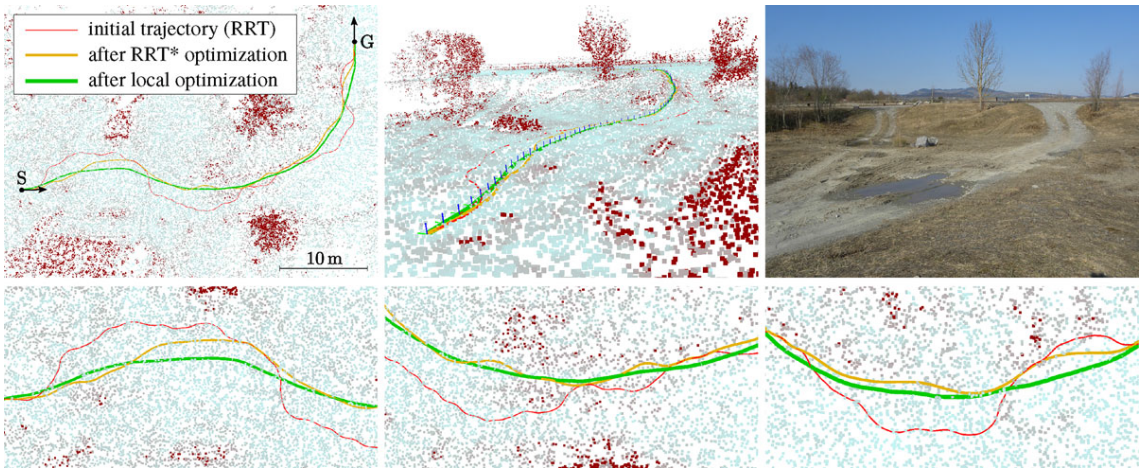
Jokaisen solmun alisolmujen yhdistäminen seuraavan solmun alisolmuihin vaatii yhdeksän liikerataa. Liikeradat esitetään kolmannen asteen kaarevuuspolynomeilla aloitusalisolmujen tasoissa. Segmenttien aloitukset ja lopetukset määritellään yhtälöiden (38), (39) ja (40) avulla siten, että jatkuvuusrajoitukset täyttyvät. Liikeradat täytyy laskea käyttäen iteratiivisia numeerisia menetelmiä, koska niille ei ole olemassa tarkkaa ratkaisua. Koska näitä liikeratoja joudutaan laskemaan huomattavan paljon jokaisella iterointikierroksella, käytetään ennalta laskettua tiheää etsintätaulua (Lookup table, LUT), jossa yhdistetään diskreetit reunaehdot – joihin kuuluvat alku- ja loppusijainti, suunta ja kaarevuus – kolmannen asteen kaarevuuspolynomin parametreihin. Itse parametreja tarvitaan vasta lopulliseen tulokseen, minkä takia muistin käytön vähentämiseksi käytetään karsittua etsintätaulua, jossa on vain maksimikaarevuus jokaiselle segmentille. Lopuksi jokaiselle liikeradan segmentille lasketaan kustannus, jotta voidaan etsiä pienimmän kustannuksen tuottava liikerata verkostosta.

Kustannukseltaan pienin liikerata etsitään verkostosta käyttämällä Dijkstran lyhimmän reitin algoritmia. Iteroinnin alussa jokaisen solmun maksimaalinen sivuttainen matka $\delta l_{max,i}$ alisolmuihin on sama. Joka kerta, kun liikerata tuottaa pienimmän kustannuksen kulkiessaan keskimmäisen alisolmun kautta, pienennetään tätä matkaa hieman, kunnes pienin mahdollinen arvo on saavutettu. Optimointi lopetetaan, kun kaikkien solmujen maksimaalinen sivuttainen matka on saavuttanut pienimmän asetetun arvon ja kaikista pienimmän kustannuksen tuottava liikerata kulkee jokaisen solmun keskimmäisestä alisolmusta. Kuvassa 35 havainnollistetaan liikkeen suunnittelun kolmen päävaiheen tuottamia reittejä avoimessa epätasaisessa maastossa.

Kuvien punainen viiva on RRT:n tuottama alkuliikerata, keltainen RRT*:llä optimoitu liikerata ja vihreä viiva lopullinen liikerata. Liikeradan jokaisen solmun asennon z-akseli, jota merkitään pienellä sinisellä viivalla, on kohtisuorassa liikeradan ja maaston kanssa. Kuvan karttapisteet on väritetty paremman havainnollistamisen vuoksi maaston epätasaisuuden mukaan. Tummanpunaiset esittävät kulkukelvottomia kohtia. Algoritmi toimii myös erittäin komplekseissa ympäristöissä. Esimerkiksi kaltevuusrajoitteet voidaan poistaa, jos robotti kykenee kulkemaan väärinpäin magneettisten pyörien avulla metallirakenteen pinnalla.



Kuva 34. *Jos solmujen välinen etäisyys on liian suuri tai pieni, liikerataan pitää lisätä tai siitä pitää poistaa solmuja. Kun etäisyys on liian suuri, uusi solmu lisätään solmujen puoleenväliin. Kun etäisyys on liian pieni, poistetaan solmun $\mathcal{N}_{i+1}$ lisäksi myös seuraava solmu ja lisätään vasta sen jälkeen uusi solmu solmujen $\mathcal{N}_i$ ja $\mathcal{N}_{i+3}$ puoleenväliin. Solmut projisoidaan tasolle $\mathcal{N}_i$. [21]*



Kuva 35. *Kuvissa havainnollistetaan liikesuunnittelualgoritmin jokaisen vaiheen tuottama tulos kahdesta eri perspektiivistä ja lisäksi mukaan on liitetty kuva maastosta. [21]*

3.3.9 Maaston arviointi – asennon laskeminen

Tässä alaluvussa tarkastellaan tarkemmin DPC-menetelmän maaston arviointia, joka noudattaa aikaisemmin määriteltyjä rajoituksia. Lähimmän maaston pinnalla sijaitsevan asennon laskeminen tapahtuu funktiolla f_{ta}^{p} seuraavasti annetulle asennolle T_{MR}. Koska annettu asento voi olla kaukanakin pinnasta, ei käytetä tiettyä etäisyyttä kyselyasennon origosta vaan tiettyä määrää K lähimpiä pisteitä. Lähimpien pisteiden määrä K riippuu kartan tiheydestä ja ajoneuvon koosta siten, että ajoneuvon peittämä alue on suunnilleen samankokoinen kuin lähimpien pisteiden muodostama alue. Tämän vuoksi voidaan päätellä, että pinnan normaali voidaan estimoida lähimpien pisteiden avulla käyttämällä pääkomponenttianalyysia (Principal component analysis, PCA). Merkitään K lähimmän pisteen indeksejä kyselypisteelle $\boldsymbol{t}_{MR}$ merkinnällä $\mathcal{N}_{\mathrm{K}}(\boldsymbol{t}_{MR})$ kartasta $\mathcal{M} = ((\boldsymbol{p}, \boldsymbol{d}_{obs})_i)_{i=1}^{N_p}$. Lasketaan näiden lähimpien pisteiden massakeskipiste ja kovarianssimatriisi:

$$\overline{\boldsymbol{p}} = \frac{1}{K} \sum_{k \in \mathcal{N}_{\mathrm{K}}(\boldsymbol{t}_{MR})} \boldsymbol{p}_k \quad \text{ja} \tag{47}$$

$$Cov(\boldsymbol{t}_{MR}) = \sum_{k \in \mathcal{N}_{\mathrm{K}}(\boldsymbol{t}_{MR})} (\boldsymbol{p}_k - \overline{\boldsymbol{p}})(\boldsymbol{p}_k - \overline{\boldsymbol{p}})^T \in \mathbb{R}^{3\mathrm{x}3}. \tag{48}$$

Pinnan normaali ilman merkkiä saadaan kovarianssimatriisin pienintä ominaisarvoa vastaavasta ominaisvektorista $\boldsymbol{v}_0$. Normaalin merkin määrittämiseen voidaan käyttää kyselyasentoa ennakkotietona:

$$\boldsymbol{n} = sign(\hat{\boldsymbol{z}}_{MR} \cdot \boldsymbol{v}_0)\boldsymbol{v}_0. \tag{49}$$

Koska suunta ei muutu kyselyasennon ja tulosasennon välillä, tulosasennon x-akseli on kyselyasennon y-akselin kanssa kohtisuorassa. Lisäksi tiedetään, että laskettu pinnan normaali on tulosasennon z-akseli. Tulosasennon x-akseli ja rotaatiomatriisi voidaan laskea seuraavasti:

$$\hat{\boldsymbol{x}}_{M\tilde{R}} = \frac{\hat{\boldsymbol{y}}_{MR} \times \mathbf{n}}{\|\hat{\boldsymbol{y}}_{MR} \times \mathbf{n}\|} \quad \text{ja} \tag{50}$$

$$\boldsymbol{R}_{M\tilde{R}} = [\hat{\boldsymbol{x}}_{M\tilde{R}} \quad \boldsymbol{n} \times \hat{\boldsymbol{x}}_{M\tilde{R}} \quad \boldsymbol{n}]. \tag{51}$$

Maakosketuspiste maastossa saadaan paikallisesti approksimoidun tason, joka on kohtisuorassa normaalin $\boldsymbol{n}$ kanssa ja sisältää massakeskipisteen $\overline{\boldsymbol{p}}$, ja kyselyasennon z-akselin leikkauspisteestä seuraavasti:

$$\boldsymbol{t}_{M\tilde{R}} = \boldsymbol{t}_{MR} + \frac{(\overline{\boldsymbol{p}} - \boldsymbol{t}_{MR}) \cdot \boldsymbol{n}}{\hat{\boldsymbol{z}}_{MR} \cdot \boldsymbol{n}} \hat{\boldsymbol{z}}_{MR}. \tag{52}$$

Tason approksimointi on voimassa vain massakeskipisteen läheisyydessä, minkä takia pitää asettaa yläraja-arvo massakeskipisteen ja maakosketuspisteen väliselle etäisyydelle. Jos etäisyys on liian suuri, asento on kulkukelvoton. Muuten etsitty maaston pinnalla oleva ajoneuvon 6D-asento saadaan seuraavasti:

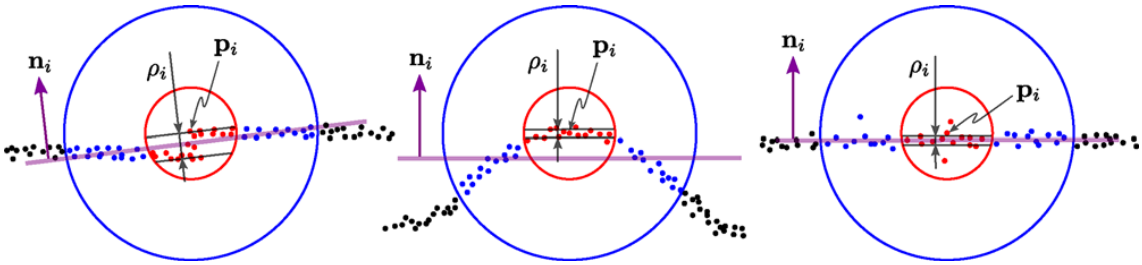
$$\boldsymbol{T}_{M\tilde{R}} = \begin{bmatrix} \boldsymbol{R}_{M\tilde{R}} & \boldsymbol{t}_{M\tilde{R}} \\ 0 & 1 \end{bmatrix}. \tag{53}$$

3.3.10 Maaston arviointi – kulkukelpoisuuden laskeminen

Kulkukelpoisuuden laskemisen funktiossa f_{ta}^{τ} käytetään kahta eri tekijää: ajoneuvon orientaatiota suhteessa painovoimaan ja maaston epätasaisuutta. Ajoneuvon orientaatio saadaan suoraan jo lasketusta $\boldsymbol{T}_{M\tilde{R}}$-matriisista, koska kartat esitetään painovoiman mukaisessa koordinaatistossa. Pinnan epätasaisuuden arvon yksittäiselle karttapisteelle tulisi olla hyvin määritelty ja yksiselitteinen kaikissa pisteissä riippumatta ajoneuvon parametreista. Yksittäisen karttapisteen epätasaisuus lasketaan vain tarvittaessa hyödyntäen läheisiä karttapisteitä, jolloin ei tuoteta ylimääräisiä keinotekoisia jaotteluita. Ajoneuvon tietyn asennon epätasaisuus maastossa on keskiarvo kaikkien niiden karttapisteiden epätasaisuuksista, jotka jäävät hieman ajoneuvoa suuremman tilavuuden sisälle. Näin esteet voidaan havaita varmemmin ja turvallisuus paranee.

Kuvassa 36 esitetään yksittäisen karttapisteen epätasaisuuden laskeminen. Ensiksi lasketaan massakeskipiste ja approksimoitu pinnan normaali aikaisemmin esitetyllä tavalla kaikille niille pisteille, jotka ovat r_{plane}-säteisen pallon sisällä annetun karttapisteen $\boldsymbol{p}_i$ ympärillä. Jotta maaston epätasaisuudet eivät tasoittuisi pois ja sileitä epäsäännöllisiä maastoja ei luokiteltaisi kulkukelvottomiksi, määritellään pienempi säde r_{res}. Epätasai-

suuden arvon laskemiseen käytetään vain niitä pisteitä, jotka ovat tämän pienemmän pallon sisällä. Näiden pisteiden indekseistä käytetään merkintää $\mathcal{B}_{\text{res}}(\boldsymbol{p}_i)$.



Kuva 36. *Pisteen $\boldsymbol{p}_i$ maaston epätasaisuuden laskeminen aloitetaan approksimoimalla maaston pintaa paikallisesti tasolla, jonka normaali $\boldsymbol{n}_i$ lasketaan kaikkien tietyn säteen r_{plane} sisällä olevien pisteiden avulla (sininen ympyrä). Epätasaisuuden arvo ρ_i lasketaan käyttämällä vain pienemmän pallon sisällä olevia pisteitä (punainen ympyrä) ja etsimällä suurin mahdollinen normaalin suuntainen askel. Osa pisteistä jätetään pois mahdollisen kohinan takia. [21]*

Näiden pisteiden etumerkillinen etäisyys tasoon lasketaan seuraavasti:

$$d_{ik} = (\overline{\boldsymbol{p}}_i - \boldsymbol{p}_k) \cdot \boldsymbol{n}_i, \qquad k \in \mathcal{B}_{\text{res}}(\boldsymbol{p}_i). \tag{54}$$

Tämän luvun varianssi kuvaisi hyvin epäjatkuvuuksia pinnan tasaisuudessa, ellei pistepilvi sisältäisi kohinaa. Kohinan takia jätetään huomioimatta määrätty murto-osa $f_\eta < 1$ pisteistä, joiden etäisyys tasoon on liian suuri. Määritetään arvo N_i seuraavasti:

$$N_i = ceil\left(\frac{f_\eta |\mathcal{B}_{\text{res}}(\boldsymbol{p}_i)|}{2}\right), \tag{55}$$

missä $|\mathcal{B}_{\text{res}}(\boldsymbol{p}_i)|$ on pisteiden määrä pienemmän säteen sisällä. Ulkopuolelle jätetään N_i suurimman ja pienimmän d_{ik} arvon omaavaa pistettä ja merkitään tätä joukkoa merkinnällä $\mathcal{O}_{\text{res}}(\boldsymbol{p}_i)$. Epätasaisuuden arvo ρ_i lasketaan suurimpana pinnan normaalin suuntaisena etäisyytenä kahden minkä tahansa jäljellä olevan pisteen välillä seuraavasti:

$$\rho_i = \left|\max_k(d_{ik}) - \min_k(d_{ik})\right|, \qquad k \in \mathcal{B}_{\text{res}}(\boldsymbol{p}_i) \backslash \mathcal{O}_{\text{res}}(\boldsymbol{p}_i). \tag{56}$$

Maaston epätasaisuus ρ_{cub} tietylle ajoneuvon asennolle $T_{M\tilde{R}}$ lasketaan kaikkien niiden pisteiden epätasaisuuksien avulla, jotka sijaitsevat ajoneuvon kokoisen turvamarginaalein laajennetun kuutiomaisen tilan $\boldsymbol{d}_{rob} \in \mathbb{R}^3$ sisällä. Näiden pisteiden indeksejä merkitään $\mathcal{C}_{\text{rob}}(\boldsymbol{T}_{M\tilde{R}})$. Kulkukelvottomina esteinä pidetään niitä pisteitä, joiden arvot ovat suurempia kuin ajoneuvokohtainen epätasaisuuden kynnysarvo ρ_{max}. Jotta esteet eivät paisuisi, pisteet, joiden epätasaisuudeksi ρ_i saadaan aluksi suurempi kuin määritetty kynnysarvo ρ_{max}, määritellään esteiksi vain silloin, kun ne kuuluvat ulkopuolelle jätettyyn joukkoon $\mathcal{O}_{\text{res}}(\boldsymbol{p}_i)$. Tällöin myös seuraava ehto pisteen etäisyydelle d_{ii} tasoon täyttyy:

$$d_{ii} \leq \min_k(d_{ik}) \vee d_{ii} \geq \max_k(d_{ik}), \qquad k \in \mathcal{B}_{\text{res}}(\boldsymbol{p}_i) \backslash \mathcal{O}_{\text{res}}(\boldsymbol{p}_i). \tag{57}$$

Muuten tämä piste merkitään kulkukelpoiseksi suurimmalla sallitulla epätasaisuudella $\rho_i = \rho_{max}$. Jos minkä tahansa pisteen epätasaisuus tässä kuutiomaisessa tilassa on suurempi kuin ρ_{max}, ajoneuvon asento merkitään kulkukelvottomaksi. Jos kaikkien pisteiden epätasaisuus on alle maksimikynnysarvon, saadaan asennon epätasaisuus kaikkien tilavuuteen kuuluvien pisteiden epätasaisuuksien keskiarvona:

$$\rho_{cub} = \frac{1}{|\mathcal{C}_{\text{rob}}(\boldsymbol{T}_{M\tilde{R}})|} \sum_{m \in \mathcal{C}_{\text{rob}}(\boldsymbol{T}_{M\tilde{R}})} \rho_m \in [0, \rho_{max}]. \tag{58}$$

Lisäksi lasketaan asentomatriisista $T_{M\tilde{R}}$ kallistuskulma ψ ja nousukulma θ, joita vertaillaan sallittuihin maksimi- ja minimiarvoihin. Sallitut kallistuskulmat oletetaan symmetrisiksi, jolloin itseisarvon tarkastelu riittää $|\psi| \leq \psi_{max}$. Ajoneuvo voi kyetä erilaisiin nousu- ja laskukulmiin, jolloin tarkastellaan, onko kulma välillä $\theta_{min} \leq \theta \leq \theta_{max}$. Jos arvot eivät pysy rajojen sisällä, asento luokitellaan kulkukelvottomaksi $\tau = 0$. Muuten kulkukelpoisuuden arvo saadaan painotetulla summalla maaston epätasaisuudesta sekä kallistus- ja nousukulmasta, jotka kaikki on normalisoitu maksimiarvoillaan:

$$\tau = 1 - \omega_{rough}\frac{\rho_{cub}}{\rho_{max}} - \omega_{roll}\frac{|\psi|}{\psi_{max}} - \omega_{pitch}\max\left(\frac{\theta}{\theta_{min}}, \frac{\theta}{\theta_{max}}\right) \in [0,1]. \tag{59}$$

3.3.11 Maaston arviointi – monikerroksiset ympäristöt

Koska maaston arvioinnissa käytetään lähimpiä pisteitä sen ominaisuuksien paikalliseen tutkimiseen ja osa näistä pisteistä voi esimerkiksi kuulua alemman kerroksen kattoon eikä tarkasteltavan kerroksen lattiaan, voivat saadut tulokset voivat olla vääriä monikerroksisissa ympäristöissä. Eri kerroksien pisteiden erottaminen tapahtuu käyttämällä hyväksi havainnointivektoria, joka kertoo, mistä suunnasta kyseinen karttapiste on havaittu. Kuva 37 havainnollistaa havainnointivektorin käyttöä eri kerrosten pisteiden erottamisessa. Paikallinen normaali $\boldsymbol{n}_i^{surf}$ lasketaan käyttämällä pientä määrää lähimpiä pisteitä ja orientoimalla se havainnointivektorin $\boldsymbol{d}_{obs,i}$ suuntaisesti siten, että seuraava pitää paikkaansa:

$$\boldsymbol{n}_i^{surf} \cdot \boldsymbol{d}_{obs,i} \overset{!}{>} 0. \tag{60}$$

Kun lasketaan maaston pinnalla sijaitsevaa asentoa, valitaan K lähimmästä pisteestä vain ne, joiden paikallinen pinnan normaali muodostaa pienemmän kulman kuin $\pi/2$ kyselyasennon z-akselin kanssa, seuraavasti:

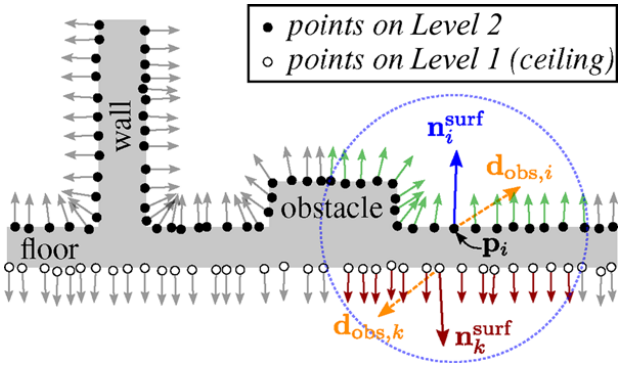
$$\hat{\boldsymbol{z}}_{MR} \cdot \boldsymbol{n}_i^{surf} \overset{!}{>} 0. \tag{61}$$

Kun estimoidaan maaston epätasaisuutta karttapisteessä $\boldsymbol{p}_i$, huomioidaan lähellä oleva piste $\boldsymbol{p}_k$ vain, jos sen paikallinen pinnan normaali muodostaa pienemmän kulman kuin $\pi/2$ pisteen $\boldsymbol{p}_i$ normaalin kanssa, seuraavasti:

$$\boldsymbol{n}_i^{surf} \cdot \boldsymbol{n}_k^{surf} \overset{!}{>} 0. \tag{62}$$

Näillä menetelmillä voidaan pisteiden erottelu tehdä paikallisesti, jolloin vältytään globaaleilta kartoitusoperaatioilta, kuten pisteiden ryhmittämiseltä tai tasojen erottamiselta.

3.3.12 Liikkeen ohjaaminen



Kuva 37. *Monikerroksisissa ympäristöissä joudutaan erottamaan eri pintojen pisteet toisistaan, kun käytetään maaston arvioinnin funktioita, jotta vältytään mahdollisesti vääriltä tuloksilta. Hyödyntämällä havainnointisuuntia $\boldsymbol{d}_{obs}$ voidaan paikalliset normaalit $\boldsymbol{n}^{surf}$ orientoida osoittamaan poispäin pinnasta. Kuvan esimerkissä jätetään pinnan epätasaisuuden laskennasta pois kaikki ne pisteet, joiden paikalliset normaalit muodostavat suuremman kuin 90°:n kulman pisteen $\boldsymbol{p}_i$ paikallisen normaalin kanssa. [21]*

Liikkeen ohjaaminen koostuu kolmesta pääosasta DPC-menetelmässä [21]. Sen pääosat ovat säädin reitinseurannalle, turvakerros ajoneuvon pysäyttämiselle ennen välitöntä törmäystä ja pysähtymisen jälkeinen palautumistoiminto. Ympäristössä tapahtuu ajallisesti nopeita ja hitaita muutoksia. Hitaat muutokset otetaan automaattisesti huomioon päivittämällä alikarttoja ja suunnittelemalla uudelleen liikeradat. Liikkeen suunnittelussa otetaan huomioon vain staattisiksi luokitellut karttapisteet. Reitinseurannan säädin pyrkii seuraamaan laskettua reittiä. Turvakerroksessa robotin reitti lasketaan tiettyyn aikahorisonttiin saakka käyttäen sekä staattista karttaa että uusinta korjattua pistepilveä. Robotti pysäytetään, jos törmäys on tapahtumassa tässä aikaikkunassa. Jos robotti on pysähtynyt, palautumistoiminto pyrkii palauttamaan sen tilaan, jossa liikkeen suunnittelu pystyy laskemaan uuden reitin.

Liikeratoja pyritään seuraamaan laskemalla ohjaukset lineaariselle nopeudelle ja kulmanopeudelle. Säädin on epälineaarinen, ja se käyttää linearisoitua tilasäätöä. Ohjauksen sisäänmenoina ovat sivuttainen virhe liikeradasta sekä lähimmän pisteen suhteen virhe suunnassa ja kaarevuudessa. Lineaarista nopeutta pienennetään vähitellen, kun sivuttainen virhe tai liikeradan kaarevuus kasvaa. Kulmanopeuden laskemisessa käytetään laskettua lineaarista nopeutta, sivuttaista virhettä, suunnan virhettä ja kaarevuutta. Päämää-

rää lähestyttäessä sallittua maksiminopeutta pienennetään ja päämäärään lisätään pieni suorasegmentti, jottei säätimen toiminnassa tapahdu mitään ei-toivottua.

Turvakerroksessa simuloidaan robotin liikkeitä nykyisten nopeuksien ja tulevien ohjearvojen perusteella käyttäen staattisia karttapisteitä ja uusinta täyden kierroksen korjattua pistepilvidataa. Aikahorisontti riippuu ajoneuvon nopeudesta ja jarrutusominaisuuksista. Ajoneuvolle määritellään maksiminopeudet ja -kiihtyvyydet sekä lisäksi viive Δt_d, jonka jälkeen halutut muutokset alkavat näkyä nopeuksissa. Iterointikierroksien maksimimäärä lasketaan aikahorisontin ja edellä mainitun viiveen avulla. Iterointi tapahtuu seuraavasti:

1. Simuloidaan robotin asentoa eteenpäin 2D-tasossa viiveen verran käyttäen senhetkisiä tai seuraavilla iterointikierroksilla laskettuja nopeuksia.
2. Käytetään maaston arvioinnin funktioita f_{ta} robotin asennon laskemiseen maaston pinnalla ja kulkukelpoisuuden laskemiseksi. Jos maastossa on este, pysäytetään robotti.
3. Lasketaan uudet nopeuden ohjaussuureet reitinseurannan säätimen avulla käyttäen simuloitua robotin tilaa.
4. Päivitetään senhetkiset nopeudet noudattaen maksiminopeuksia ja -kiihtyvyyksiä seuraavasti:

$$\begin{aligned} &\Delta v = \min(|v_{ctrl} - v_{cur}|, \dot{v}_{max}\Delta t_d)\, sign(v_{ctrl} - v_{curr}), \\ &\Delta\omega = \min(|\omega_{ctrl} - \omega_{curr}|, \dot{\omega}_{max}\Delta t_d)\, sign(\omega_{ctrl} - \omega_{curr}), \\ &\mathrm{v_{curr}} \leftarrow \min(v_{curr} + \Delta v, v_{max}), \\ &\omega_{\mathrm{curr}} \leftarrow \min(|\omega_{\mathrm{curr}} + \Delta\omega|, \omega_{max})\, sign(\omega_{\mathrm{curr}} + \Delta\omega). \end{aligned} \tag{63}$$

Jos simuloinnissa ei tapahdu törmäystä, käytetään ensimmäisellä kierroksella laskettuja nopeuksien ohjaussuureita.

Kun robotti ei voi liikkua eteenpäin ja se joutuu pysähtymään, palautumistoiminto käskee robotin liikkumaan hitaasti taaksepäin. Navigointi jatkaa normaalia toimintaansa vasta, kun liikkeen suunnittelija onnistuu laskemaan jälleen uuden reitin.

3.3.13 Algoritmin muokkausehdotuksia runko-ohjattaville ajoneuvoille kaivosympäristössä

Tässä alaluvussa esitellään tarvittavia muutoksia, joita edellä esitetty algoritmi mahdollisesti tarvitsee toimiakseen runko-ohjattaville ajoneuvoille kaivosympäristössä. Merkittävin muutoksia aiheuttava tekijä on ajoneuvo, etenkin muuttuvan rungon takia. Toinen tekijä on ympäristö, joka mahdollistaa erilaisten valintojen tekemisen liikesuunnittelussa.

Ajoneuvo vaikuttaa ainakin ajoneuvon asennon arviointiin, kulkukelpoisuuden laskentaan ja alkeisliikeratojen muodostukseen. Edellä esitetyssä algoritmissa ajoneuvon run-

gon asento ja renkaiden muodostama pinta ovat koko ajan lähestulkoon vakioita, kun taas runko-ohjattavissa ajoneuvoissa koko ohjausjärjestelmä perustuu siihen, että nämä ominaisuudet muuttuvat. Tämän takia muutoksia tulee tehdä maaston arviointiin kuuluvaan ajoneuvon asennon laskentaan, johon vaikuttaa hyvin suuresti keskinivelen kulman suuruus. Tietenkin, jos voidaan olettaa tienpinta hyvin tasaiseksi, voidaan käyttää samaa menetelmää kuin aikaisemminkin. Eri ajoneuvo vaikuttaa myös kulkukelpoisuuden arviointiin. Jos esimerkiksi ajoneuvon alla olisi iso kuoppa tai kumpare, johon renkaat tai runko eivät kuitenkaan osuisi missään vaiheessa, aikaisemmin esitetty algoritmi saattaisi määrittää kohdan kulkukelvottomaksi. Lisäksi yksinkertaiset kolmannen asteen kaarevuuspolynomit alkeisliikeratojen muodostamisessa eivät ole riittäviä runko-ohjattaville ajoneuvoille. Kaarevuuspolynomit runko-ohjattaville ajoneuvoille on käsitelty tarkemmin jo luvussa 2.6.

Runko-ohjattavan ajoneuvon asennon arviointi tapahtuu seuraavasti hyödyntäen samoja menetelmiä kuin aikaisemmin:

1. Arvioidaan paikallinen pinnan normaali suunnilleen ympyrämäiseltä alueelta, jonka halkaisija on ajoneuvon akselin leveys, ja tarkistetaan sen perusteella, ylitetäänkö ajoneuvolle asetettuja staattisen asennon turvarajoja.
2. Tämän jälkeen yritetään etsiä pisteet, jotka ovat mahdollisesti renkaiden lähellä, käyttämällä suorakulmaisia laatikoita, jotka ovat hieman renkaita suuremmat.
3. Koska ajoneuvon taka-akselissa on pieni oskillaatiovara, otetaan keskiarvo takarenkaiden pisteiden sijainnista.
4. Käytetään takarenkaiden keskiarvoa ja eturenkaiden lähistöllä olevia pisteitä ajoneuvon akseleiden keskipisteiden maanpinnalle projisoitujen asentomatriisien laskemiseen.
5. Tarkistetaan tämän jälkeen vielä staattisen asennon turvarajat ja lisäksi, sisältääkö ajoneuvon runko tai vaihtoehtoisesti turvaraja pisteitä eli onko mahdollisia törmäyksiä tässä asennossa.
6. Lopuksi voidaan laskea kulkukelpoisuuden vaatima epätasaisuus renkaiden lähistöllä olevien pisteiden avulla.

Tälläkin menetelmällä tehdyllä asennon arvioinnilla ja näihin liitetyillä alkeisliikeradoilla tulee olemaan virheitä etenkin, jos tienpinta on erittäin epätasainen ja renkaiden liukukulmat poikkeavat nollasta. Toisena huomiona voisi nostaa sen, että tämä ei tarkasta kuin kyseisen asennon mahdolliset törmäykset. Liikeratojen solmujen välisissä tiloissa olevat törmäykset saattavat jäädä vielä havaitsematta etenkin suurilla keskinivelen kulmilla.

Seuraavaa menetelmän avulla voidaan ainakin testata, onko jokin piste konveksin kappaleen sisällä. Tällöin seuraavan ehdon täytyy täyttyä:

$$\widehat{\boldsymbol{n}}_i \cdot \boldsymbol{p}_i \underset{\forall i}{\leq} 0, \tag{64}$$

missä $\widehat{\boldsymbol{n}}_i$ on jokin konveksin kappaleen pinnan normaali, joka osoittaa kappaleesta poispäin, ja $\boldsymbol{p}_i$ on jokin vektori kyseisen pinnan jostakin pisteestä tarkasteltavaan pisteeseen. Jos yksikin pistetulo on positiivinen, piste on konveksin kappaleen ulkopuolella. Myös muita geometrian sääntöjä voidaan mahdollisesti hyödyntää konveksin kappaleen muodostamisessa, kuten palloja tai lieriöitä eli pisteen etäisyyttä pisteestä tai suorasta. Tällöin pitää vain tarkastella ehdot tarkemmin tapauskohtaisesti.

Liikeratojen solmujen välisiä törmäyksiä voitaisiin tarkastella esimerkiksi muodostamalla yksi konveksi kappale ottamalla tasaisin välimatkoin yksittäisiä rungon asentoja turvarajoineen ja yhdistämällä nämä toisiinsa. Sisäkaarteen puolelta pitää tarkastelusta poistaa erikseen konveksi kappale, jotta vältetään tarkasteltavan törmäyskappaleen paisuminen. Maanpinta kannattaa olettaa tasaiseksi yksittäisen liikeradan sisällä tarkastelun helpottamiseksi. Turvarajoja voisi olla myös eri tasoja, esimerkiksi kriittinen, erittäin lähellä ja lähistöllä. Näitä voisi hyödyntää myös liikesuunnittelussa esimerkiksi reitin ryhmittämiseksi tunnelin oikeaan reunaan. Ajoneuvon törmäystarkasteltava konveksi kappale voidaan myös jakaa eri segmentteihin, jolloin liikesuunnittelua on mahdollista ohjata tarkastelemalla mahdollisia törmäyksiä näihin segmentteihin. Jos suurimmalla turvarajalla määritelty konveksi kappale ei sisällä ulkopuolisia pisteitä, eli se on törmäyksetön, ei pienemmilläkään turvarajoilla törmäyksiä ole.

Liikesuunnittelussa voidaan tehdä erilaisia valintoja, koska kaivosympäristö on hyvin suljettu ja ainakin vielä lähitulevaisuudessa liikenne siellä on ylemmällä tasolla valvottua. Jos esimerkiksi tiedetään, että tunneli on yksisuuntainen tai yksikaistainen eikä siellä ole muuta liikennettä, voidaan perusreitti keskittää tunnelin kulkusuuntaan nähden. Lisäksi liikenteen valvonnan takia useita eri tunnelivaihtoehtoja sisältävät reitit lähtö- ja maalipisteen välillä eivät välttämättä ole liikesuunnittelun käytössä. Näiden syiden vuoksi aikaisemmin esitetyn liikesuunnittelualgoritmin toinen vaihe eli RRT* on oikeastaan merkityksetön, koska se ei voi tehdä merkittäviä parannuksia reittiin rikkomatta asetettuja liikennesääntöjä.

Laskennallisia hyötyjä voidaan saavuttaa käyttämällä tehokkaasti alikarttoja ja niiden muodostamaa verkostoa. Alikarttojen verkoston avulla voidaan ohjata ensimmäisen vaiheen RRT:n hakua oikeaan suuntaan. Lisäksi tunnelin keskialue on helpommin laskettavissa alikartoista kuin prosessoimalla kokonaista pistepilveä tämän muodostamisen jälkeen. Käyttämällä vain lähimpiä alikarttoja voidaan mahdollisesti myös laskennallisesti raskaahko kerrosten erottaminen jättää väliin. Tunneleissa voi olla joitain kriittisiä pisteitä, joissa ei ole paljon sivuttaissuuntaista liikkumavaraa kaivoskoneille. Näissä kohdissa algoritmilla voi olla ongelmia etenkin, jos alkeisliikkeet on diskretoitu liian harvoiksi tai niistä ei voi muodostaa tunneliin sopivia yhdistelmiä. Näihin kohtiin voisi olla hyvä asettaa jollakin menetelmällä tunnelitopologian solmu, josta voidaan jatkaa seuraavaan segmenttiin.

Liikesuunnittelun lyhyiden matkojen liikeratojen muodostamisessa tulee huomioida myös mahdolliset projektioista johtuvat muutokset liikeradan kaarevuuteen ja sitä kautta myös muutokset keskinivelen kulmaan. Jos esimerkiksi horisontaalisessa tasossa oleva liikerata projisoidaan osittaiseen ylämäkeen ja käytetään samaa keskinivelen asentoa kuin horisontaalisessa tasossa, taka-akselin keskipiste voi olla eri kohdassa kuin projisoitu reitti etenkin silloin, kun keskinivelen kulma ei ole nolla.

Liikesuunnittelun viimeiseen vaiheeseen eli liikeradan paikalliseen optimointiin tulee vielä tehdä muutoksia, jotta se huomioisi runko-ohjattavan ajoneuvon ominaisuudet. Esimerkiksi liikeratojen alisolmujen kaarevuuden ja siten myös keskinivelen asennon määrittäminen ei onnistu enää aikaisemmin esitetyllä tavalla. Suunnan määritys voi onnistua samoin, mutta sitä käyttämällä reittiin voi tulla epäoptimaalisia kaarroksia. Liikeradan paikallinen optimointi pitää luultavasti suorittaa iteratiivisesti. Se voisi tapahtua esimerkiksi seuraavasti:

1. Siirretään sivuttaissuuntaisesti verkon solmuja siten, että pidetään suunta samana mutta kaarevuutta kasvatetaan sisämutkaan ja pienennetään ulkomutkaan siirrettävän matkan ja alkuperäisen kaarevuuden mukaan. Näille kohdille tehdään maaston arviointi ja törmäyksien tarkastelu turvarajoineen. Turvarajojen segmenttien avulla voi olla mahdollista muuttaa sivuttaissuuntaista siirtymismatkaa jo tässä vaiheessa.
2. Valitaan sitten verkosta lyhin törmäyksetön reitti, jonka solmukohtiin lasketaan suunnat kuten perustapauksessa käyttäen yhtä ympyrän kaarta.
3. Tämän jälkeen kaarevuuden arvo pyritään minimoimaan verkon solmuissa. Lisäksi solmujen ja näiden välisten tilojen täytyy olla törmäyksettömiä. Kaarevuuden derivaatan arvo on hyvä myös pitää mahdollisimman pienenä.
4. Jos reitti on vieläkin törmäyksetön mahdollisten reitin tasoitusoperaatioiden jälkeen, jatketaan samoin kuin aiemmin esitellyssä paikallisessa optimoinnissa. Mahdolliset törmäykset reitillä voidaan korjata käyttämällä eri alisolmua tai hyödyntämällä turvarajojen segmenttejä sivuttaissuuntaisessa siirtymisessä.

Liikeradan paikallinen optimointi vaatii kuitenkin vielä lisätutkimusta, etenkin reitin tasoitusoperaatio, jotta reitti ei sisällä liikaa käännöksiä tai äkkinäisiä liikkeitä. Lisäksi olisi hyvä perehtyä siihen, kuinka reitin keskittäminen tai sijoittaminen tunnelin reunaan onnistuisi parhaiten. Paikallisen optimoinnin menetelmiä voisi mahdollisesti käyttää tulevaisuudessa myös perusreittien pieneen varioimiseen tienpinnan kulumisen estämiseksi. Liikesuunnitteluun kuuluu myös nopeusprofiilin muodostaminen, johon ei tässä työssä keskitytä.

3.4 BLAM – avoimen lähdekoodin SLAM-algoritmi lasertutkille

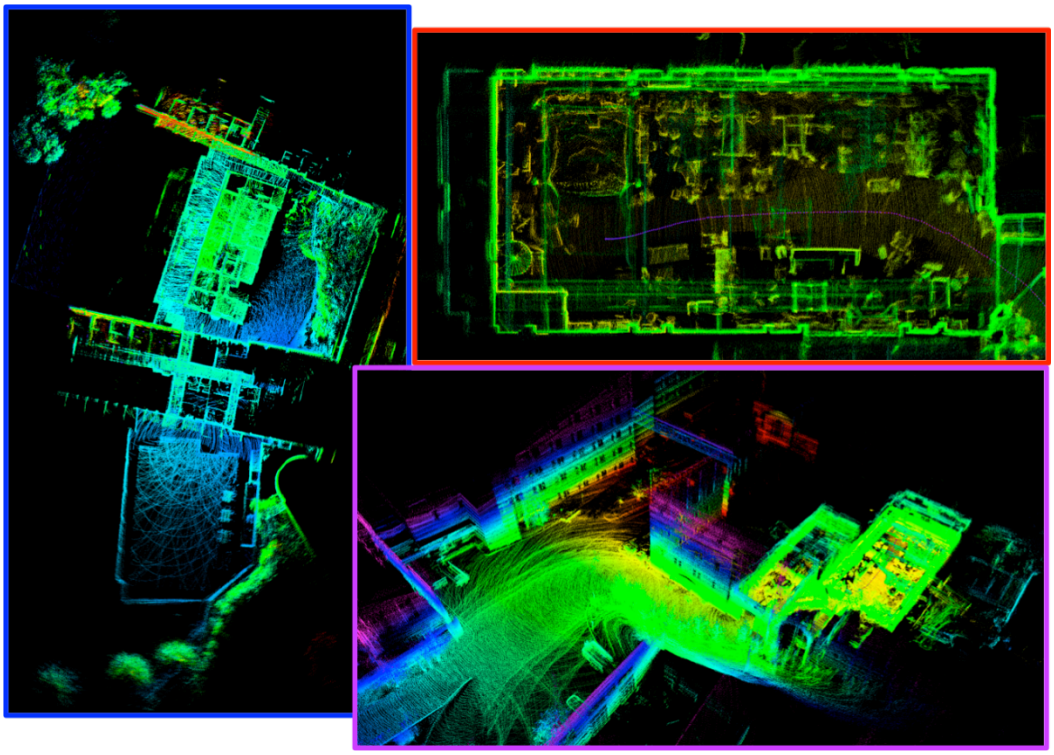
BLAM (Berkeley Localization And Mapping) on avoimen lähdekoodin ohjelmistopaketti reaaliaikaiseen 3D-paikannukseen ja kartoitukseen lasertutkan avulla. Sen on ke-

hittänyt Nelson Berkeleyn tekoälyn tutkimuslaboratoriossa (BAIR Laboratory). Avoin lähdekoodi on saatavilla lähteestä [34]. Tämän työn pistepilvikartta muodostetaan BLAM-algoritmin avulla eikä DPC-menetelmän SLAM-algoritmin avulla juuri avoimen lähdekoodin takia. BLAM-algoritmissa on kuitenkin useita puutteita verrattuna tutkimusryhmän algoritmiin. Näistä puutteista kerrotaan myöhemmin lisää. Todelliset järjestelmät kannattaa toteuttaa mukaillen tutkimusryhmän artikkelia.

BLAM-algoritmin toiminnasta ei ole olemassa erillistä dokumentaatiota, joten tässä työssä käydään läpi sen perustoiminta. Myöhemmin käsitellään algoritmin puutteita.

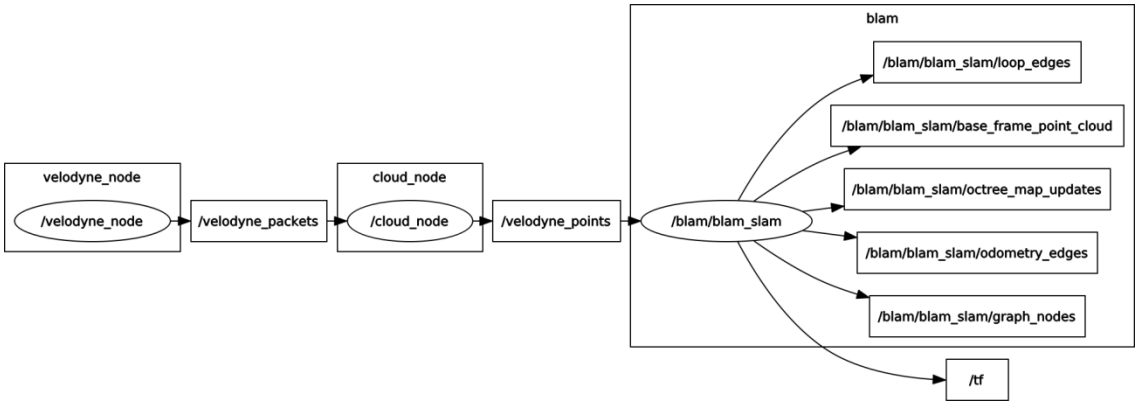
3.4.1 BLAM-algoritmin yleiskuvaus

BLAM-algoritmi käyttää GTSAM-kirjastoa (Georgia Tech Smoothing and Mapping) ja ROS:ia (Robot Operating System). GTSAM käyttää Boost-kirjastoa, joka vaatii ROS:lta vähintään Indigo-version, joka vuorostaan toimii ainakin Ubuntu 14.04 -versiossa. Kuvassa 38 esitetään algoritmin muodostamia pistepilvikarttoja. Oikean yläkulman kuvassa on havainnollistettu myös anturin kulkema reitti. Pisteiden väri riippuu korkeudesta. Koordinaatisto ei kuitenkaan välttämättä ole painovoiman mukainen, koska pistepilvien muodostamisessa käytetään pelkästään lasertutkalla saatua dataa, joten koordinaatisto riippuu alkuorientaatiosta.



Kuva 38. *BLAM-algoritmi muodostaa pistepilvikarttoja lasertutkan datan perusteella. Algoritmi tallentaa myös anturin kulkeman reitin. Algoritmin toiminta perustuu ICP:hen. [35]*

Lasertutkana käytetään sekä tässä työssä että algoritmin kehittäjän esimerkeissä Velodynen VLP-16:ta, joka tuottaa 16-kanavaisia täyden kierroksen skannauksia ympäristöstä noin 10 Hz:n taajuudella. Riippuen laitteen toimintatilasta tämä tarkoittaa noin 300 000 3D-pistettä sekunnissa. Itse algoritmi perustuu ICP:hen kuten myös DPC-menetelmässä käytetty algoritmi. Kuvassa 39 on visualisoitu ROS-solmujen laskentaverkko, johon kuuluu BLAM-solmun lisäksi Velodynen pcap-tiedostoa lukeva solmu ja kuolleet kulmat poistava pilvisolmu, joka julkaisee yksittäisen kokonaisen pistepilven BLAM-solmulle.



Kuva 39. *Velodyne_node toistaa pcap-tiedostosta lähes reaaliaikaisesti VLP-16-anturin tuottamat paketit, joista cloud_nodessa poistetaan kuolleet kulmat ja muodostetaan BLAM:lle julkaistava yksittäinen kokonainen pistepilvi.*

3.4.2 BLAM-algoritmin toiminta

BLAM koostuu pistepilvien kartoittajasta, paikantajasta, silmukansulkijasta, odometriasta ja suodattimesta sekä mittausten synkronoijasta. Yhtenä muutoksena koodiin tässä työssä on tehty havainnointinormaalien laskenta DPC-menetelmän innoittamana. Koodin käyttämä PCL-kirjasto tukee onneksi myös 3D-pisteitä, joilla on normaalit.

Kun kaikki osat on alustettu, odotetaan ensimmäistä mittausta. Ensiksi mittaukselle lasketaan havainnointinormaalit. Seuraavaksi mittausta suodatetaan runsaasti poistamalla jopa 90 % pisteistä. Jos mittaus on ensimmäinen, alustetaan odometria suodatetulla mittauksella sekä kartoittaja ja silmukansulkija suodattamattomalla mittauksella ja odotetaan seuraavaa mittausta. Käyttämällä peräkkäisiä suodatettuja mittauksia päivitetään vähittäinen liikkeen estimaatti ICP:n avulla. Suodatettu mittaus muunnetaan tämän estimaatin ja integroidun estimaatin avulla maailmankoordinaatistoon. Senhetkisestä kartasta arvioidaan silloiselle mittaukselle lähimmät naapurit, jotka muunnetaan takaisin anturin koordinaatistoon. Päivitetään vähittäisen liikkeen estimaatti ja integroitu liikkeen estimaatti ICP:n avulla uudelleen tässä koordinaatistossa. Seuraavaksi tarkastetaan, sulkeutuuko jokin silmukka. Silmukan sulkeutuessa regeneroidaan kartta ja päivitetään integroitu liikkeen estimaatti. Tällöin mahdollisia geometrisia virheitä silmukan sisällä saadaan osittain korjattua. Ellei silmukka sulkeutunut, tarkastetaan vielä, voi-

daanko skannaus lisätä verkkoon avainkehyksenä. Avainkehyksien verkkoa käytetään silmukoiden sulkeutumisen tarkasteluun. Jos odometria on liikkunut yli kynnysarvon verrattuna edelliseen avainkehykseen, lisätään uusi avainkehys verkkoon ja skannauksen pisteet karttaan.

Koodissa on useita eri parametreja, jotka täytyy muuttaa sovellukseen sopiviksi. Lisäksi algoritmin suorituskyky näyttää parantuvan, jos pienennetään ROS:n julkaisijoiden ja tilaajien jonojen kokoja yhteen. Jos koot ovat suurempia ja algoritmilla kestää käsitellä jotain mittausta liian kauan, vanhimmat mittaukset putoavat pois ja peräkkäin käsiteltävien mittausten välille syntyy epäjatkuvuuksia, jolloin liikkeen estimaattien tarkkuus heikkenee. Lasertutkan tuottama mittausdata on tallennettu aikaisemmin tiedostoon, jota ajetaan tässä työssä mahdollisimman reaaliaikaisesti todellisen järjestelmän toiminnan simuloimiseksi. Lisäksi skannauksen kuolleet pisteet, jotka muodostuvat ajoneuvosta itsestään, kannattaa poistaa tiedostoa ajettaessa, ellei halua karttaan virhemittauksia.

3.4.3 BLAM-algoritmin puutteet

Algoritmissa on useita puutteita esimerkiksi verrattuna DPC-artikkelin SLAM-algoritmiin. Ensinnäkin siinä ei oteta huomioon skannauksen sisäistä virhettä, joka syntyy pyörivien lasertutkien liikkeestä yhden mittauskierroksen aikana. Toiseksi se ei huomioi dynaamisia kohteita mitenkään, joten karttaan voi jäädä artefakteja näistä kohteista. Se ei myöskään päivitä karttoja aktiivisesti muuten kuin vertaamalla, onko mittauksen piste jo lisätty aikaisemmin karttaan.

Koska järjestelmä käyttää vain lasertutkadataa, pistepilvikartta on orientoitu anturin alkuorientaation mukaisesti. Maassa kulkevien ajoneuvojen takia on suotavaa, että kartat on orientoitu painovoiman mukaisesti. Lisäksi algoritmi ei käytä ollenkaan inertiamittauksia, jolloin ICP voi samankaltaisissa ympäristöissä saavuttaa lokaalin minimin, joka ei välttämättä ole oikeassa paikassa. Nämä virheet kertautuvat karttaan ellei silmukoita saada suljettua. Jos järjestelmässä tapahtuu virhe paikannuksessa, se ei uudelleen paikanna itseään jo mitattuun karttaan vaan alkaa luoda uutta karttaa ihan vanhan kartan viereen.

3.5 Pyörillä liikkuvien robottien dynaamisten mallien muotoilu ja kalibrointi

Seegmiller esittää väitöskirjassaan [43] pyörillä liikkuville roboteille dynaamisten mallien formuloinnin, joka on erittäin tarkka, yleispätevä, modulaarinen ja nopea. Dynaamisia 3D-malleja tarvitaan, jos halutaan suunnitella, ohjata tai estimoida liikkeitä haastavissa maastoissa, tehtäessä aggressiivisia liikkeitä tai käsiteltäessä painavia kuormia. Kinemaattisissa ja dynaamisissa formuloinneissa hän hyödyntää differentiaalisia algebrallisia yhtälöitä (DAE, Differential Algebraic Equation) liikkeen ennustamiseen. Lisäk-

si hänen työssään hyödynnetään Baumgarten stabilointia rajoitetun ja ajautumattoman liikkeen ennustamiseen. Työssä ennustetaan myös renkaiden luistamista. Dynaamisessa formuloinnissa käytetään voimatasapaino-optimointia, jotta rajoitukset voidaan huomioida käytettäessä DAE-yhtälöitä, joiden avulla voidaan käyttää suurempia integrointiaskeleita verrattuna perinteisiin menetelmiin. Lisäksi hän esittää menetelmän, jolla mallien parametrit voidaan kalibroida. Työstä on saatavilla avoimet MATLAB- ja C++-kirjastot pyörillä liikkuvien robottien dynamiikkaohjelmistolle (Wheeled Mobile Robot Dynamics Engine, WMRDE).

Simulaatiot pyörivät jopa 10 tuhatta kertaa reaaliaikaista nopeammin, minkä ansiosta tämä menetelmä on hyvä vaihtoehto tarkempaan liikkeen ennustamiseen tai suunnitteluun, jos paikallinen maaston approksimointi ei ole riittävän tarkka esimerkiksi liikkeen suunnitteluun tai törmäyksen ennakoimiseen. Tässä työssä on toteutettu kaivoskoneesta malli hyödyntäen MATLAB-kirjastoa. Lisäksi lähdekoodia on muokattu siten, että simulaatio toimii myös pistepilvien päällä 2,5D-ympäristöjen lisäksi. Tällöin vältytään laskennallisesti raskaalta ympäristön pintojen rekonstruktiolta.

4. TULOKSET

Jotta pystytään suunnittelemaan liikerata mille tahansa ajoneuvolle, tarvitaan ensin ympäristömalli. Tässä työssä ympäristömallina toimii 3D-pistepilvi, jonka jokaisella pisteellä on havainnointinormaalit, ja se on luotu avoimen lähdekoodin BLAM-algoritmilla. Suunnittelussa käytetään kuitenkin hieman kevennettyä versiota pistepilvestä, josta on poistettu havainnointinormaalien avulla katot ja osa seinistä, ja lisäksi sitä on suodatettu hieman hajamittausten pois saamiseksi. Tätä 3D-pistepilveä käytetään demonstroimaan alkuliikeradan muodostamista runko-ohjattavalle ajoneuvolle DPC-menetelmän kaltaisella RRT:llä.

Aluksi tässä luvussa käsitellään ympäristömallin luominen BLAM-algoritmilla ja tarkastellaan mallin oikeellisuutta maanmittarin kiintopisteiden avulla. Lisäksi käsitellään lyhyesti valealikarttaverkoston muodostaminen, jolla voidaan nopeuttaa liikesuunnittelualgoritmia. Seuraavaksi luvussa näytetään esimerkki alkuliikeradan muodostamisesta. Lopuksi käsitellään vielä jatkotutkimuksissa mahdollisesti hyödyllistä WMRDE:llä mallinnettua ja simuloitua kaivoskonetta 2,5D-maastossa. Simulaattorin voi muokata myös toimimaan pistepilvillä, mutta silloin täytyy käyttää parempaa pyörämallia tai suodattaa kohinaa sisältävät pistepilvet kaivoskoneen hyppimisen estämiseksi.

4.1 Ympäristömallin luominen BLAM-algoritmilla

Seuraavissa alaluvuissa esitetään BLAM-algoritmin tuottama pistepilvi Sandvikin Tampereen testitunnelista. Ensimmäisessä alaluvussa kuvataan, kuinka mittaus suoritettiin. Toisessa alaluvussa tarkastellaan tarkemmin itse algoritmin tuottamaa pistepilveä ja siitä muokattu kevennettyä versiota. Sen jälkeen tarkastellaan pistepilven kolmiulotteisten mittasuhteiden oikeellisuutta hyödyntämällä maanmittarin pisteitä. Viimeisessä alaluvussa käsitellään valealikarttaverkon luominen pistepilvestä.

Algoritmissa on käytetty taulukossa 1 esitettyjä parametrien arvoja. Parametrit, joita ei ole listattu taulukkoon, ovat oletusarvoisia. Arvot on valittu siten, että käytettävissä olleen tietokoneen resurssit saadaan hyödynnettyä mahdollisimman hyvin. Tietokoneen suoritin on Intel Core i7-4710HQ, asennetun RAM-muistin määrä 16 Gt ja näytönohjain GTX980M. Ubuntu-käyttöjärjestelmä on asennettu perinteiselle 7200 RPM:n kovalevylle.

Taulukko 1. *Taulukkoon on merkitty osa BLAM-algoritmissa käytettyjen parametrien arvoista. Loput parametrit ovat oletusarvoisia.*

BLAM-osa: Parametrin nimi	Arvo
Rate: Estimate	20.0
Filtering: Grid filter	true
Filtering: Grid resolution	0.2
Filtering: Random filtering	true
Filtering: Decimate persentage	0.85
Map: Octree resolution	0.05
Localization: Correspondence distance	1
Localization: ICP iterations	2
Odometry ICP: Correspondence distance	2
Odometry ICP: ICP iterations	2
Loop closures: Check for loop closures	true
Loop closures: Translation threshold	0.5
Loop closures: Proximity threshold	0.5
Loop closures: Max tolerable fitness	0.1
Loop closures: Skip recent poses	100
Loop closures: Poses before reclosing	100

4.1.1 Ympäristömallidatan mittaaminen

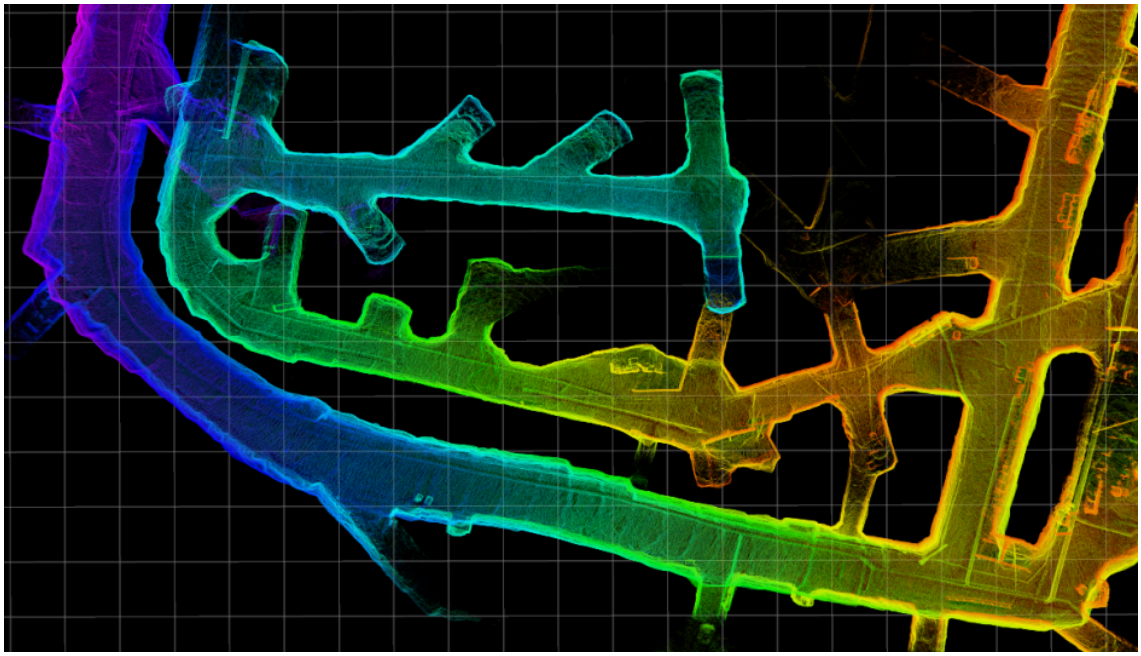
Ympäristödatan kerääminen suoritettiin Velodynen VLP-16-3D-lasertutkalla, joka oli asennettu pakettiauton katolle hieman etuviistoon kallelleen. Anturi oli asetettu yksittäisen palautuksen tilaan ja huomioimaan vain viimeinen mittaus. Pakettiautolla ajettiin Sandvikin testikaivoksessa hitaasti vajaat puoli tuntia. Reitin jokainen osa ajettiin suunnilleen kerran kumpaankin suuntaan, mutta reitti ei sisältänyt fyysisen sijainnin perusteella yhtään silmukkaa. Tosin yhden silmukan voi mahdollisesti muodostaa näköyhteyden avulla.

BLAM-algoritmin toiminnan kannalta oli erittäin hyvä, että mittaus suoritettiin ajamalla hitaasti, koska algoritmissa ei ole huomioitu virheitä, jotka syntyvät anturin liikkeestä yhden skannauskierroksen aikana. Mittauksen tuottamat UDP-paketit tallennettiin tiedostoon, jota on ajettu tässä työssä suunnilleen reaaliaikaisesti tuottamaan 3D-pistepilvi testikaivoksesta. Osa anturin säteistä osui systemaattisesti pakettiautoon, minkä takia kyseiset pisteet on poistettu pistepilvistä ennen kuin nämä on syötetty BLAM-algoritmille.

4.1.2 BLAM-algoritmin tuottama pistepilvi

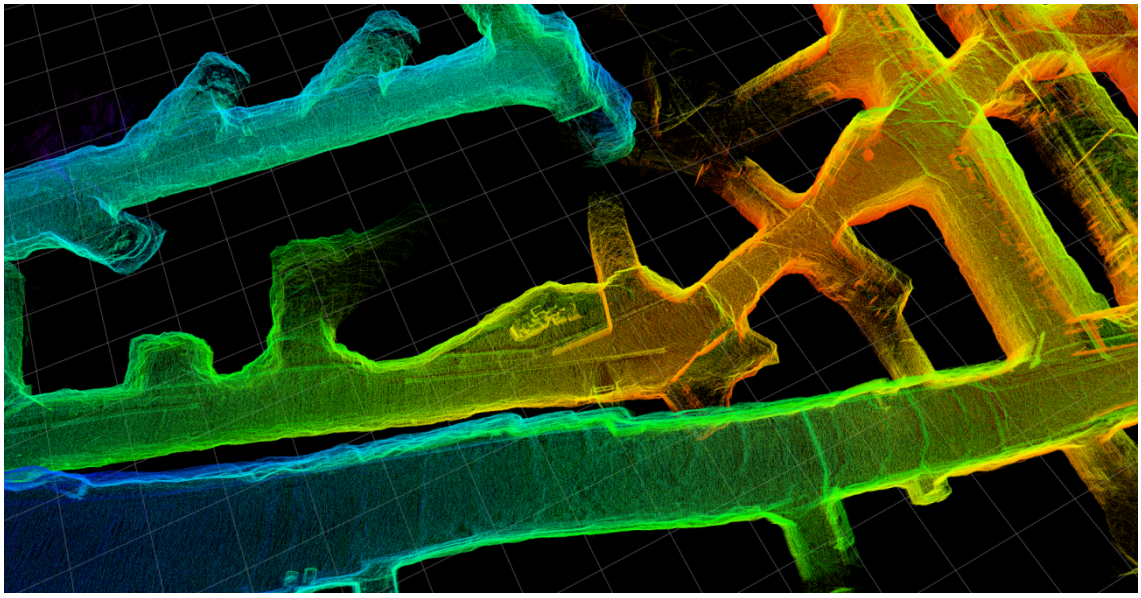
Pistepilven tallennustarkkuutena käytettiin 5 cm:n vokseleita eli pikseleiden kolmiulotteisia vastineita, jotta tiedoston koko ei kasvanut liian suureksi. VLP-16-anturin tark-

kuudeksi on luvattu ±3 cm. Kuvassa 40 on esitetty BLAM-algoritmin Sandvikin testikaivoksesta tuottama pistepilvi kokonaisuudessaan ylhäältä päin kuvattuna.



Kuva 40. *BLAM-algoritmin tuottama pistepilvi Sandvikin testikaivoksesta koostuu noin 19 miljoonasta pisteestä. Taustalla olevien haaleiden neliöiden reunojen pituus on 10 metriä. Pisteiden värit kertovat korkeuden siten, että kellertävät ja punertavat ovat matalalla ja vastaavasti sinertävät ovat korkealla. Pistepilvi on orientoitu käsin suunnilleen painovoiman mukaisesti. Kuvakaappaus on Rviz-ohjelmasta.*

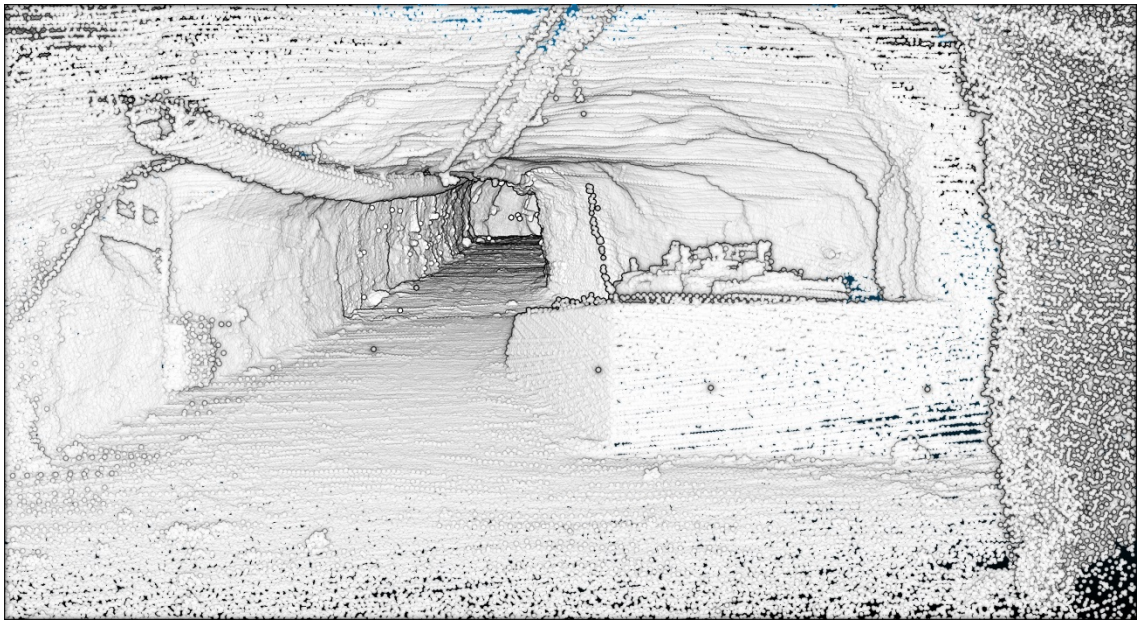
Kuvan pistepilvi koostuu noin 19 miljoonasta pisteestä, ja sen taustalla näkyvien neliöiden sivujen pituus on 10 metriä. Pisteiden värit määräytyvät korkeuden mukaan. Kuvassa 41 on lähikuva tunnelin keskiosasta.



Kuva 41. *Pistepilvestä on mahdollista havaita tarkkojakin kohteita.*

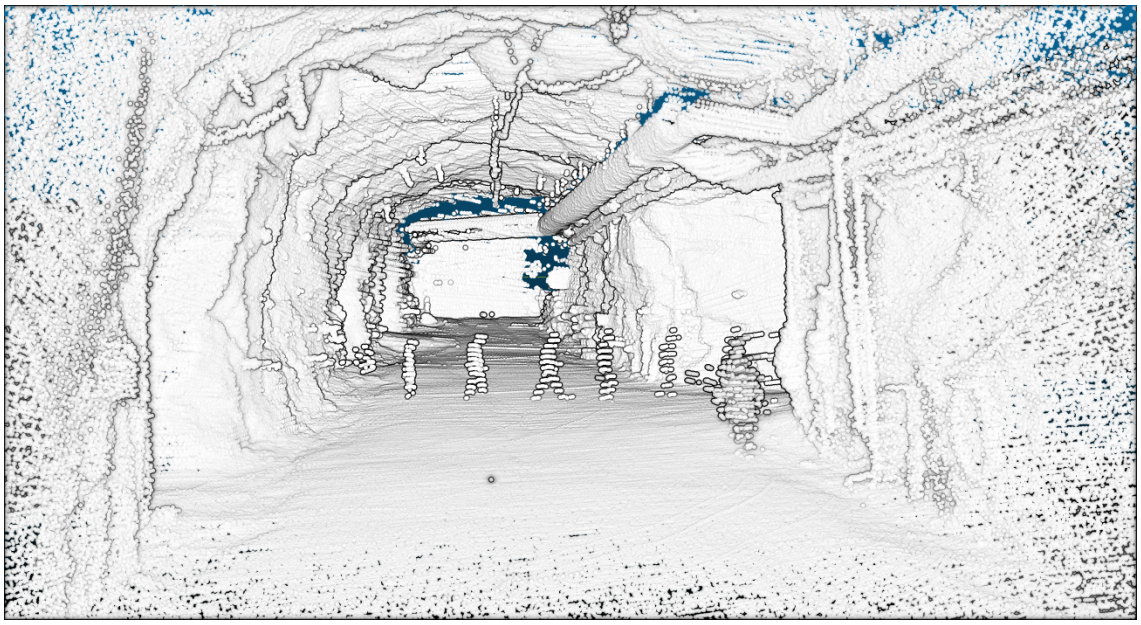
Kuvien pistepilvi on orientoitu käsin suunnilleen painovoiman mukaisesti, koska lähtötaso ei ollut painovoiman mukaisessa tasossa. Kuvista voidaan havaita monia eri yksityiskohtia, kuten putkia tunneliston katossa ja tunnelin seinän viereen pysäköityjä ajoneuvoja.

Kuvassa 42 on esitetty näkymä pistepilven sisältä. Kuvasta voidaan havaita muutamia hajapisteitä ilmassa, jotka ovat syntyneet joko ilmassa olevista hiukkasista tai ajoneuvon katon heijastuksista, jos anturi on heilahtanut hieman ajon aikana suhteessa ajoneuvoon tai jos asetetut rajat kuolleiden pisteiden poistamiseksi eivät ole olleet tarpeeksi tiukat. Tunnelin reunoilla on havaittavissa kohinaa, joka voi johtua anturin tarkkuudesta, ICP:n huonosta konvergoitumisesta tai anturin liikkeestä yhden skannauskierroksen aikana. Kuvassa näkyvä tyhjä tila kameraa lähellä olevien pisteiden välissä johtuu CloudCompare-ohjelman renderöinnistä. Lisäksi kuvassa on hyvä esimerkki lasertutkien mahdollisesta heikkoudesta joka ilmenee silloin, kun niiden horisontaalinen tai vertikaalinen tarkkuus ei ole tarpeeksi hyvä, jolloin erilaiset kapeat puomit tai johdot eivät ole välttämättä helposti havaittavissa.



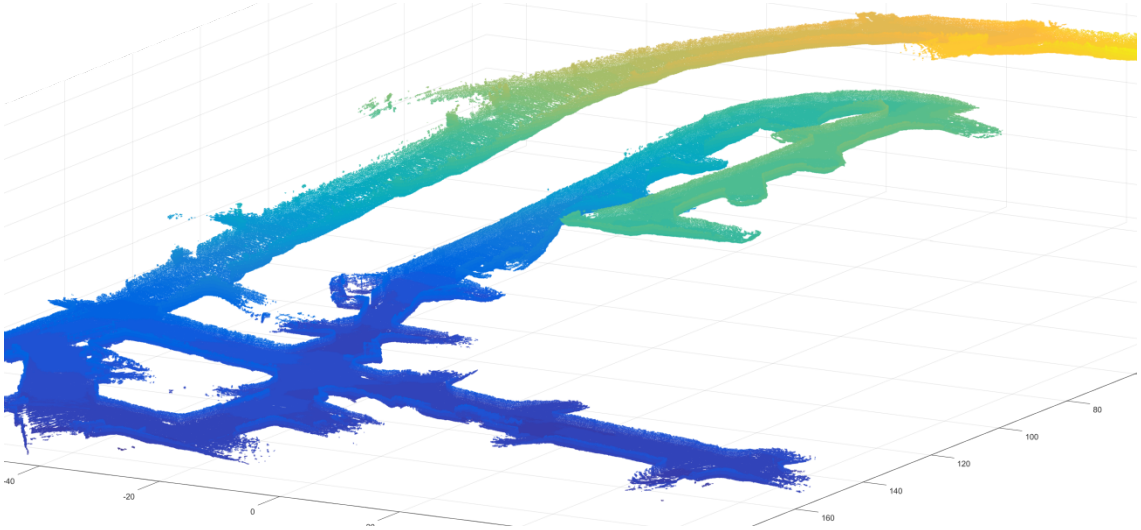
Kuva 42. *Pistepilven sisällä voidaan havaita muutamia hajapisteitä ajoneuvon katosta tai ilmassa olevista hiukkasista. Lisäksi tunnelin reunoilla voidaan nähdä hieman kohinaa, joka voi johtua anturin tarkkuudesta, ICP:stä tai anturin liikkeestä yhden skannauskierroksen aikana. Kameran lähellä olevien pisteiden välissä näyttäisi olevan enemmän tyhjää kuin kaukana olevien pisteiden välissä johtuen kuvan renderöinnistä.*

Kuvassa 43 on esimerkki BLAM-algoritmin heikkoudesta eli dynaamisista kohteista. BLAM-algoritmi ei nykymuodossaan tosiaan poista dynaamisia kohteita pistepilvestä. Kuvassa näkyy, kuinka henkilö tai henkilöt ovat kävelleet yhteensä kaksi kertaa anturin näköpiirissä. Artefaktit eivät kuitenkaan ole täysin jatkuvia johtuen tallennusmenetelmästä, joka tallensi skannatun pistepilven vain, jos tämä on avainkehys eli ajoneuvo on liikkunut tarpeeksi pitkälle edellisestä avainkehyksestä.



Kuva 43. *BLAM-algoritmi ei poista dynaamisia kohteita pistepilvestä.*

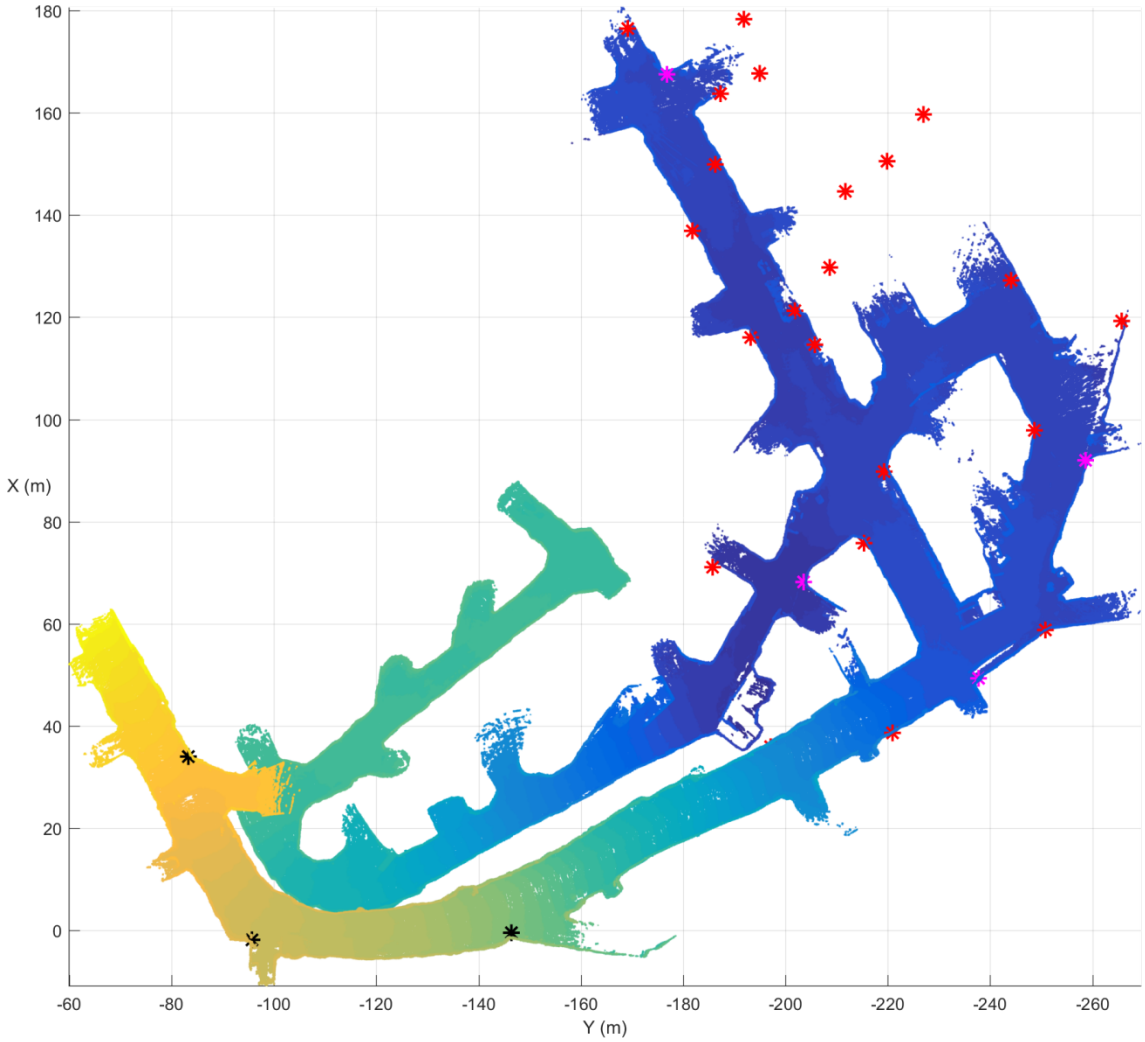
Kuvassa 44 on esitetty reitinsuunnittelussa käytetty pistepilvi. Suunnittelussa käytetään laskennan nopeuttamiseksi kevyempää mallia, josta on poistettu havainnointinormaalien avulla katto ja osa seinistä. Mallia on harvennettu siten, että siinä on enää vain 1,5 miljoonaa pistettä. Lisäksi siitä on suodatettu hajapisteitä pois.



Kuva 44. *Suunnittelussa käytetään harvempaa mallia, josta on katto ja osa seinistä poistettu. Mallia on myös suodatettu hajapisteiden poistamiseksi. Tämä malli sisältää vain 1,5 miljoonaa pistettä.*

4.1.3 Pistepilven vertailu kiintopisteiden kanssa

Kuvassa 45 on esitetty BLAM-algoritmin tuottama pistepilvi, johon on merkitty myös maanmittarin kartoittamat kiintopisteet. Selkeyden vuoksi ja kiintopisteiden näkymisen



Kuva 45. *BLAM-algoritmilla muodostettu 3D-pistepilvi, johon on merkitty tähdillä kiintopisteet. Tunneliston katon pisteet on poistettu kuvasta. Pistepilven koordinaatisto on muunnettu samaksi kuin kiintopisteiden neljän purppuran värisen pisteen avulla etsimällä silmämääräisesti näiden vastinpisteet pistepilvestä ja laskemalla sitten muunnosmatriisi. Punaiset ja purppuran väriset ovat uuden koordinaatiston kiintopisteitä ja mustat ovat vanhan, mutta koordinaatistojen välinen yhteys on tuntematon, minkä vuoksi mustat pisteet ovat luotettavasti vertailukelpoisia vain keskenään. Vanhat kiintopisteet vaikuttavat olevan oikeassa paikassa suorakulmaisella koordinaattien siirrolla. Kun verrataan silmämääräisesti kiintopisteitä pistepilven kanssa, ei havaita suuria eroja, minkä takia pistepilveä voidaan hyödyntää liikesuunnittelussa ainakin paikallisesti.*

parantamiseksi pistepilvestä on poistettu tunneliston katon pisteet. Tunnelistossa on kiintopisteitä kahdesta eri koordinaatistosta, joista vanhempi ei ole enää aktiivisessa käytössä. Uuden ja vanhan koordinaatiston välinen tarkka muunnos ei ole tiedossa, mutta vanhan koordinaatiston kiintopisteet on saatu silmämääräisesti oikeille paikoilleen pelkästään suorakulmaisella siirrolla. Tämän takia vanhan koordinaatiston kiintopisteet ovat luotettavasti vertailukelpoisia vain keskenään. Kiintopisteet on merkitty kuvaan tähdillä, joista punaiset ja purppurat ovat uudesta koordinaatistosta ja mustat ovat vanhasta koordinaatistosta. Pistepilven koordinaatisto on muunnettu uuteen koordinaatistoon etsimällä ensiksi silmämääräisesti vastinpisteet pistepilvestä purppuran värisille

kiintopisteille. Muunnosmatriisi on laskettu käyttäen lähteessä [47] esitettyä jäykän liikkeen laskemistapaa, joka hyödyntää pienintä neliösummaa ja pääakselihajotelmaa. Laskentaan voitaisiin käyttää kaikkia pisteitä, jolloin saataisiin virheelle tarkempi numeerinen arvo, mutta tämä vaatisi mahdollisimman tarkkaa kiintopisteiden manuaalista sijoittamista pistepilveen. Tämän takia muunnosmatriisin laskemiseen ja tuloksen silmämääräiseen tarkasteluun on tyydytty käyttämään vain neljää eri kiintopistettä, joille on etsitty suunnilleen vastaavat kohdat pistepilvestä.

Edellisestä kuvasta voidaan havaita, että jokainen siihen lisätty kiintopiste näyttäisi olevan lähellä seinää eikä ainakaan poikkea useita metrejä. Virheet muunnoksessa voivat johtua epätarkasta kiintopisteiden vastinpisteiden asettamisesta pistepilveen. Tietenkin pistepilvi voisi sisältää myös oikeita virheitä, mutta näillä etäisyyksillä BLAM on tuottanut kokonaisen pistepilven, jonka kolmiulotteiset mittasuhteet ovat ainakin lähellä oikeaa.

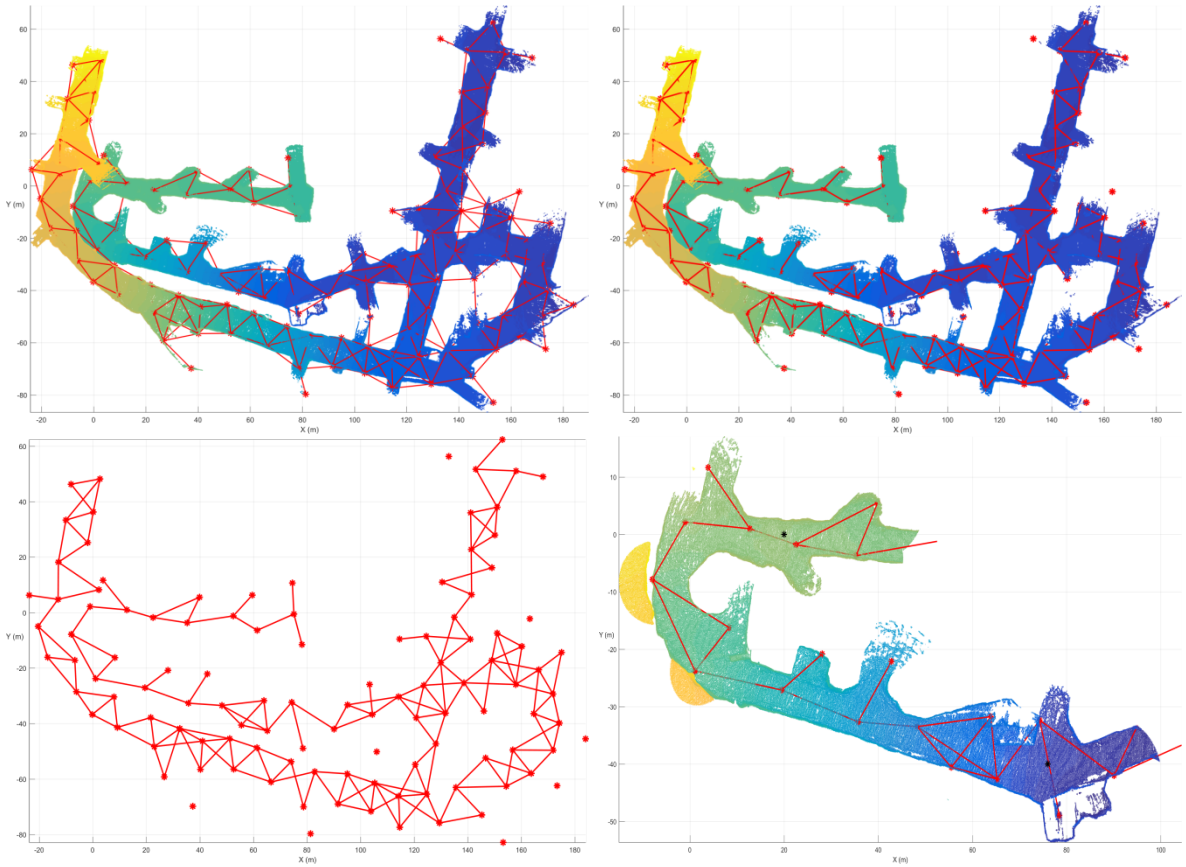
Tarkastelemalla kuvassa 45 näkyviä kiintopisteitä ja seiniä nähdään, että ainakaan näillä etäisyyksillä kokonaiseenkaan pistepilviympäristömalliin ei tule useiden metrien kokoisia virheitä. Tämä tulos on rohkaiseva ajatellen navigointia ja liikesuunnittelua etenkin siksi, että näissä vain paikallisen ympäristömallin tarkkuuden pitää olla riittävä.

4.1.4 Valealikarttaverkoston luominen

BLAM-algoritmiin ei ole tässä työssä kuitenkaan toteutettu vielä alikarttaverkoston muodostamista tai dynaamisten pisteiden poistamista. Jotta tämä työ voisi demonstroida liikesuunnittelun RRT:n ohjatun haun algoritmin toimintaa, lasketaan jälkikäteen kokonaisesta siistitystä pistepilvestä valealikarttaverkosto. Tämä tehdään seuraavasti:

1. Arvotaan pistepilvestä jokin piste, joka ei kuulu vielä mihinkään valealikarttaan, ja lisätään kaikki tietyn etäisyyden (10 m) sisällä tästä pisteestä olevat pisteet uuteen valealikarttaan, kunnes jokainen piste kuuluu ainakin yhteen valealikarttaan.
2. Lisätään reunat jokaisen valealikartan väliin, joilla on yhteisiä pisteitä ja joiden keskipisteiden välinen kulma ei nouse vaakatasosta liian suureksi ($0{,}4\pi$ rad). Tämä toimii etenkin silloin, kun tunneliston katon pisteet on poistettu.
3. Seuraavaksi tarkastetaan, onko eri valealikarttojen välissä seiniä. Tämä onnistuu projisoimalla kaikki valealikarttojen pisteet tasolle ja tarkastamalla lähimpien pisteiden etäisyydet keskipisteiden välisen suoran eri kohtiin (0,5 m:n välein). Jos yksikin etäisyys ylittää kynnysarvon (0,5 m), tulkitaan valealikarttojen välisen reunan sisältävän seinän. Tällöin reuna poistetaan. Jos seinät ovat kaltevia tai liian ohuita, menetelmä ei välttämättä tuota oikeaa tulosta.

Näin muodostetun valealikarttaverkoston avulla voidaan löytää reitti tai reittejä eri pisteiden välille käyttämällä verkkoteorioiden algoritmeja. Kuvassa 46 on esitetty esimerkki tällä tavoin muodostetusta valealikarttaverkostosta ja sen käytöstä.

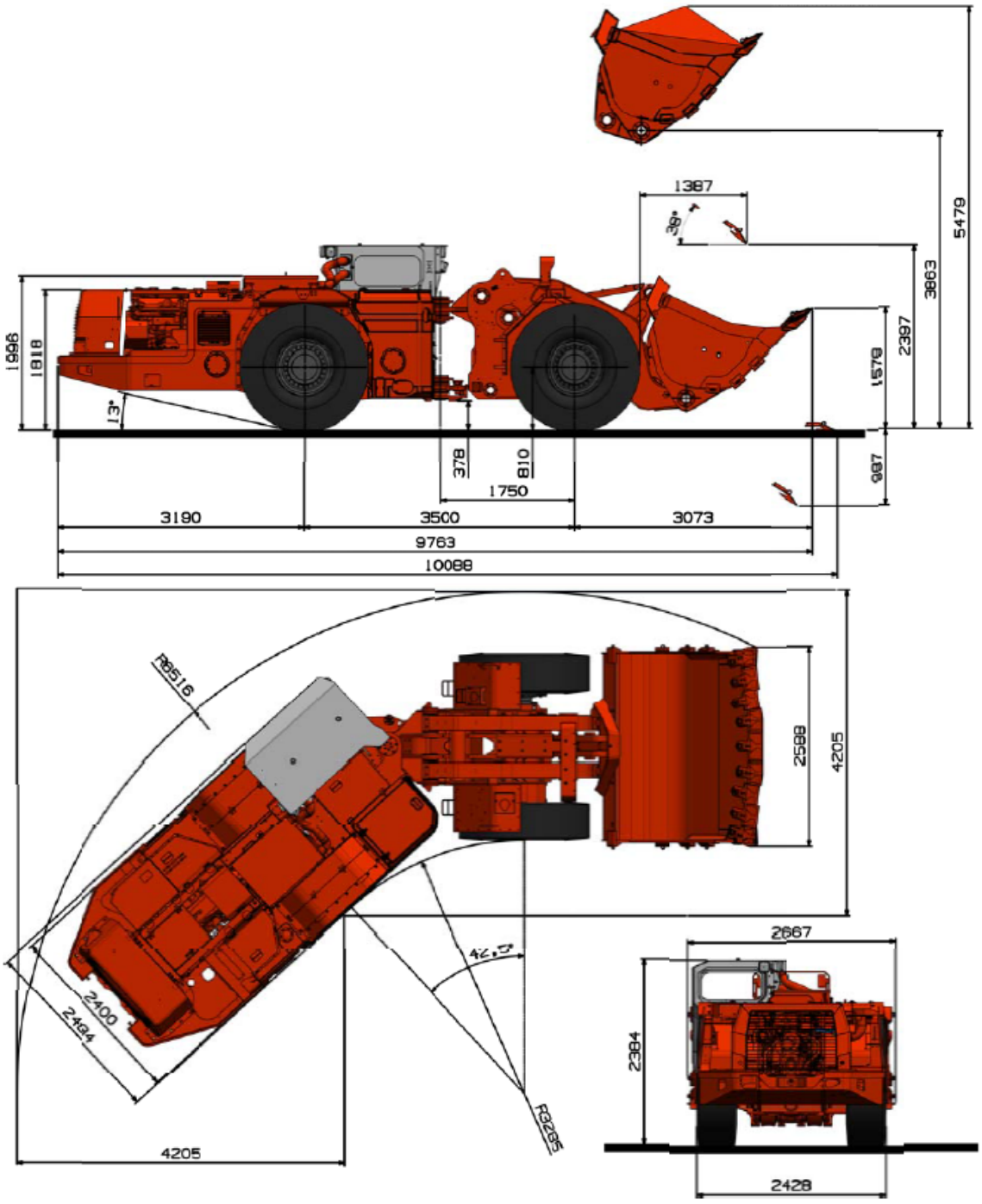


Kuva 46. *Yläkuvissa on näkyvissä koko tunneliston pistepilvi. Näistä vasemmassa on myös näkyvissä kaikki valealikarttoja yhdistävät reunat paitsi ne, joiden välinen kulma on liian suuri vaakatason kanssa. Oikeasta kuvasta on poistettu myös ne reunat, joiden välillä on seinä. Vasemmassa alakuvassa on esitetty pelkästään valealikarttaverkosto ja oikeassa alakuvassa esimerkki, kuinka verkostoa voidaan hyödyntää. Jälkimmäisen vasemmassa reunassa on havaittavissa myös ylemmän tunnelin pisteitä, joita ei ole poistettu valealikarttaverkoston luonnissa. Näitä pisteitä ei myöskään olisi aidossa alikarttaverkostossa.*

SLAM-algoritmien tulisi koota heti alikarttaverkosto pistepilven muodostuksen yhteydessä, jolloin kokoaminen olisi tehokkaampaa ja mahdollistaisi alikarttojen päivittämisen myöhemmin.

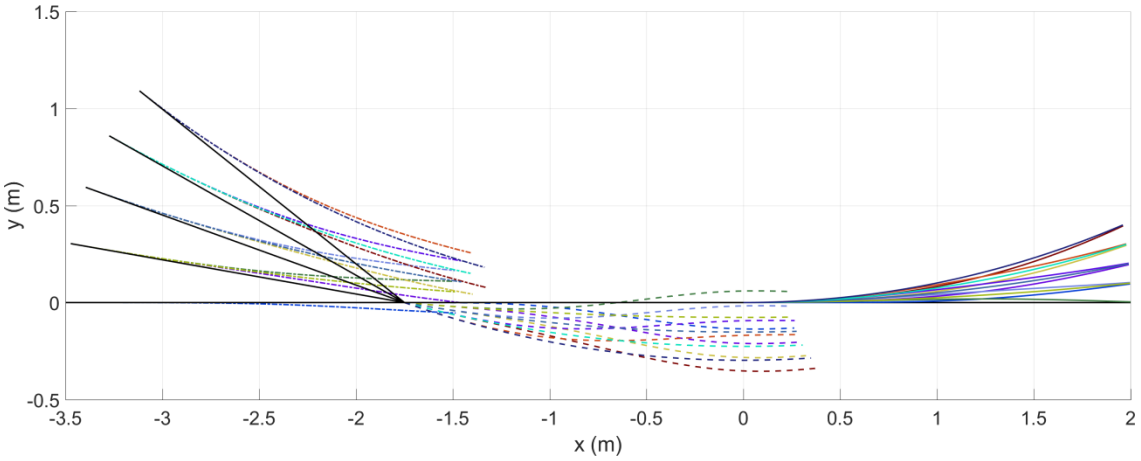
4.2 Liikesuunnittelualgoritmin testaus kaivoskoneelle

Tässä työssä liikesuunnittelualgoritmin kaivoskoneen parametreina käytetään datalehdeltä [40] löytyviä arvoja, joiden perusteella on myös arvioitu puuttuvat arvot. Kuvassa 47 on kaivoskoneen dimensiot.

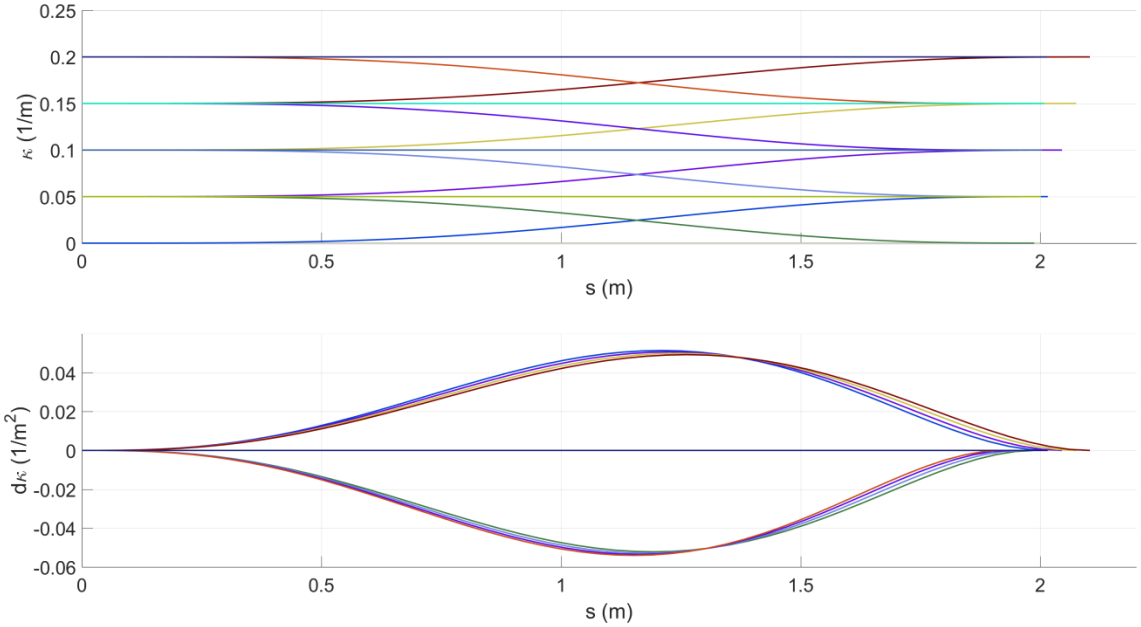


Kuva 47. *Työssä käytetyt kaivoskoneen dimensiot ja muut tekniset tiedot on saatu tai arvioitu datalehdestä. [40]*

Kuvaan 48 on koottu RRT:n laajentamisessa käytetyt tasoliikeradat ja kuvaan 49 on merkitty näiden kaarevuus ja derivaatta matkan funktiona.



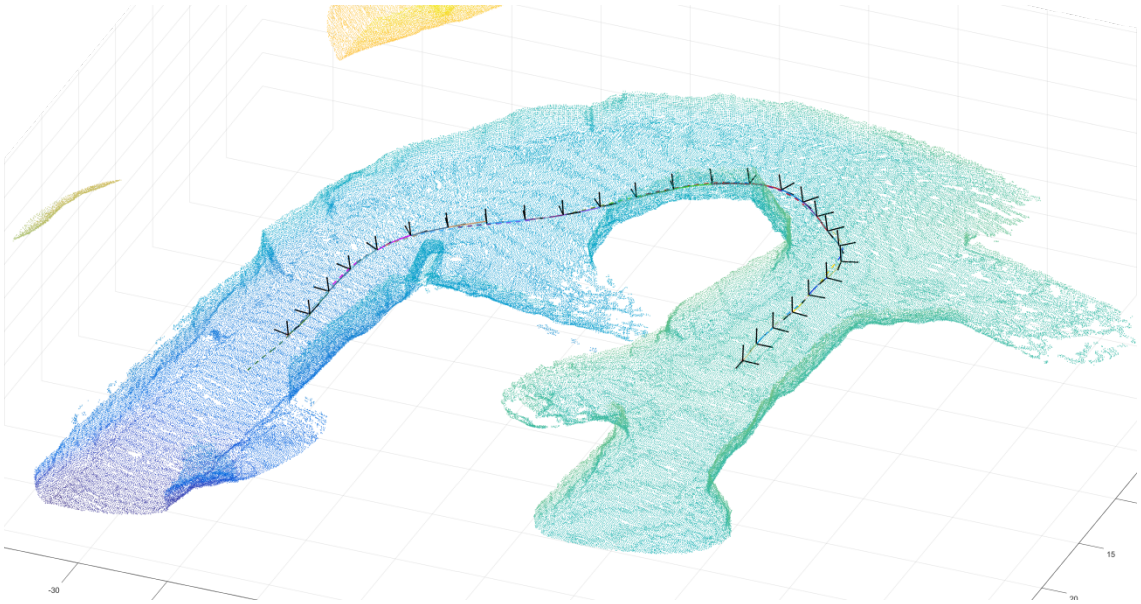
Kuva 48. *RRT:n laajentamisessa käytetyt tasoliikeradat on tässä työssä diskretoitu kuvan mukaisesti. Kaivoskoneen keskinivelen aloituskulmiksi on valittu viisi eri asentoa. Jokaisesta liikeradan keskinivelen kulmasta päästään joko viereiseen kulmaan tai kulma pidetään samana koko liikeradan ajan. Liikeradat voidaan täydentää symmetrian avulla x-akselin mukaan. Yhtenäiset viivat esittävät etuakselin keskipistettä, katkoviivat keskiniveltä ja pistekatkoviivat taka-akselin keskipistettä.*



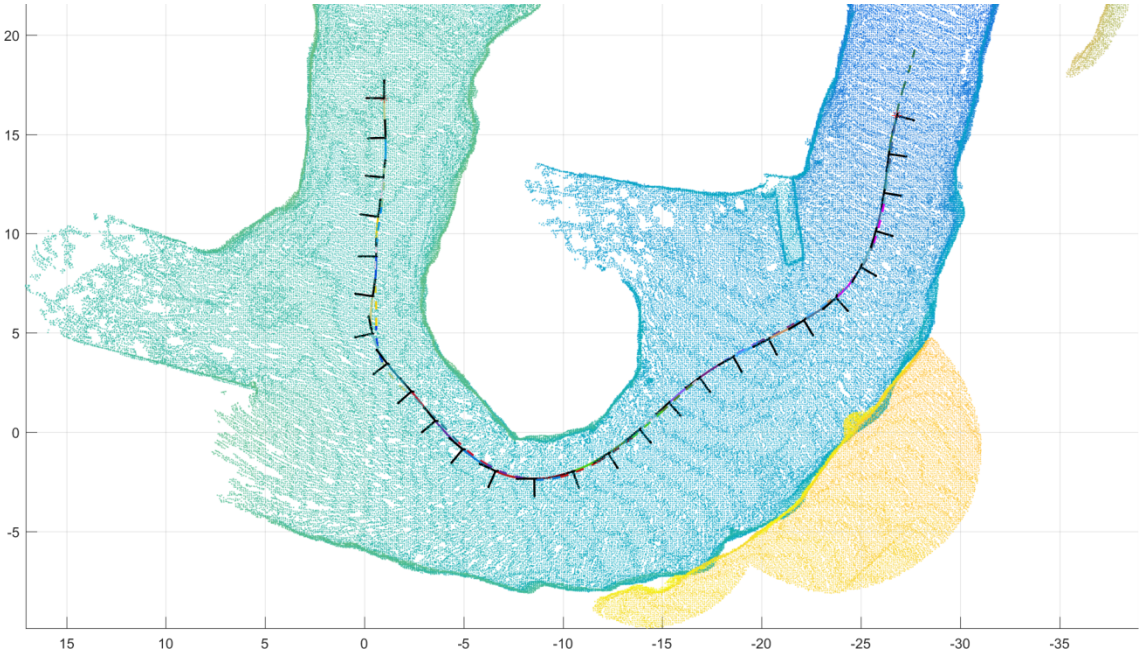
Kuva 49. *Tasoliikeratojen kaarevuuden ja sen derivaatan muutos matkan suhteen pidetään pienenä. Lisäksi derivaatta on nolla liikeradan alussa ja lopussa.*

Tasoliikeradat on muodostettu 2 metrin säteelle siten, että niiden kaarevuuden ja oletuksen mukaisesti myös keskinivelen derivaatta on nolla alku- ja loppupisteessä. Lisäksi keskinivelen muutosta on rajoitettu siten, että se ei voi muuttua muuhun kuin vierekkäiseen diskretoituun asentoon tai pysyä koko ajan samana, ja lisäksi muutokset tapahtuvat monotonisesti. Kaarevuuspolynomina on käytetty kuudetta astetta ja asteet ensimmäisestä kolmanteen on asetettu nolliksi.

Kuvissa 50 ja 51 on esitetty RRT:n muodostama alkuliikerata kaivoskoneelle.



Kuva 50. *RRT:llä muodostettu alkuliikerata kaivoskoneelle ilman RRT*:ä, paikallista optimointia ja seinien etäisyyksien huomioimista on kohtalainen. Yhtenäiset viivat edustavat etuakselin keskikohdan liikettä maanpinnalle projisoituna ja vastaavasti katkoviivat taka-akselin. Kaivoskoneen etuakselin maanpinnalle projisoitu keskikohta eri solmuille on merkitty mustilla akseleilla.*



Kuva 51. *Alkuliikerata kuvattuna ylhäältä näyttää, että reitti kulkee hieman liian läheltä seinää.*

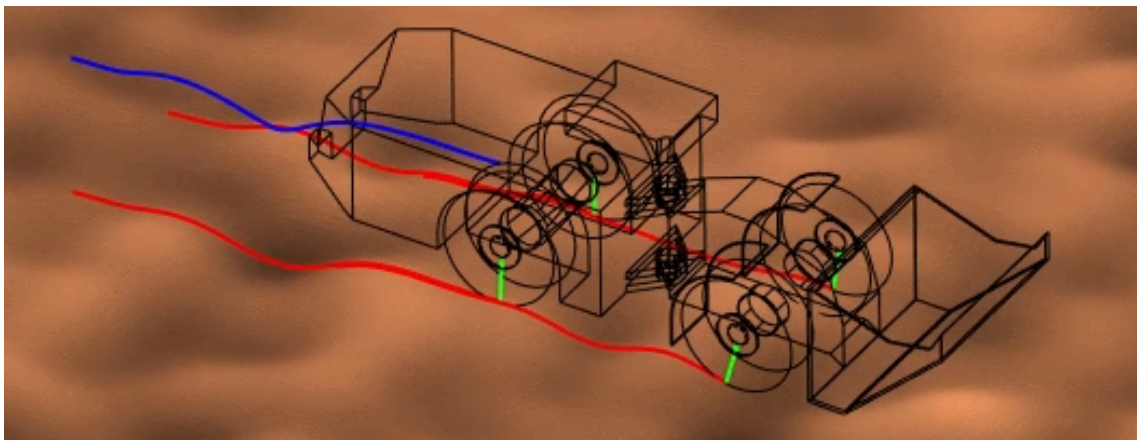
Tummat akselit kuvissa edustavat etuakselin keskipistettä solmuissa projisoituna maan pinnalle. Yhtenäiset viivat esittävät etuakselin keskipisteen liikettä ja vastaavasti katkoviivat taka-akselin. Etu- ja taka-akseli kulkevat lähes samoja reittejä paitsi lopun läheisyydessä, jossa RRT:n puut on yhdistetty ja keskinivelen kulmaa muutetaan nopeammin. Reitti kulkee hyvin läheltä seinää ja saattaa sen vuoksi sisältää törmäyksen kaivos-

koneen kauhan kanssa. Nimenomaan solmujen väliset törmäykset tulisi tarkastaa vielä erikseen etenkin suurten kaarevuuksien läheisyydessä.

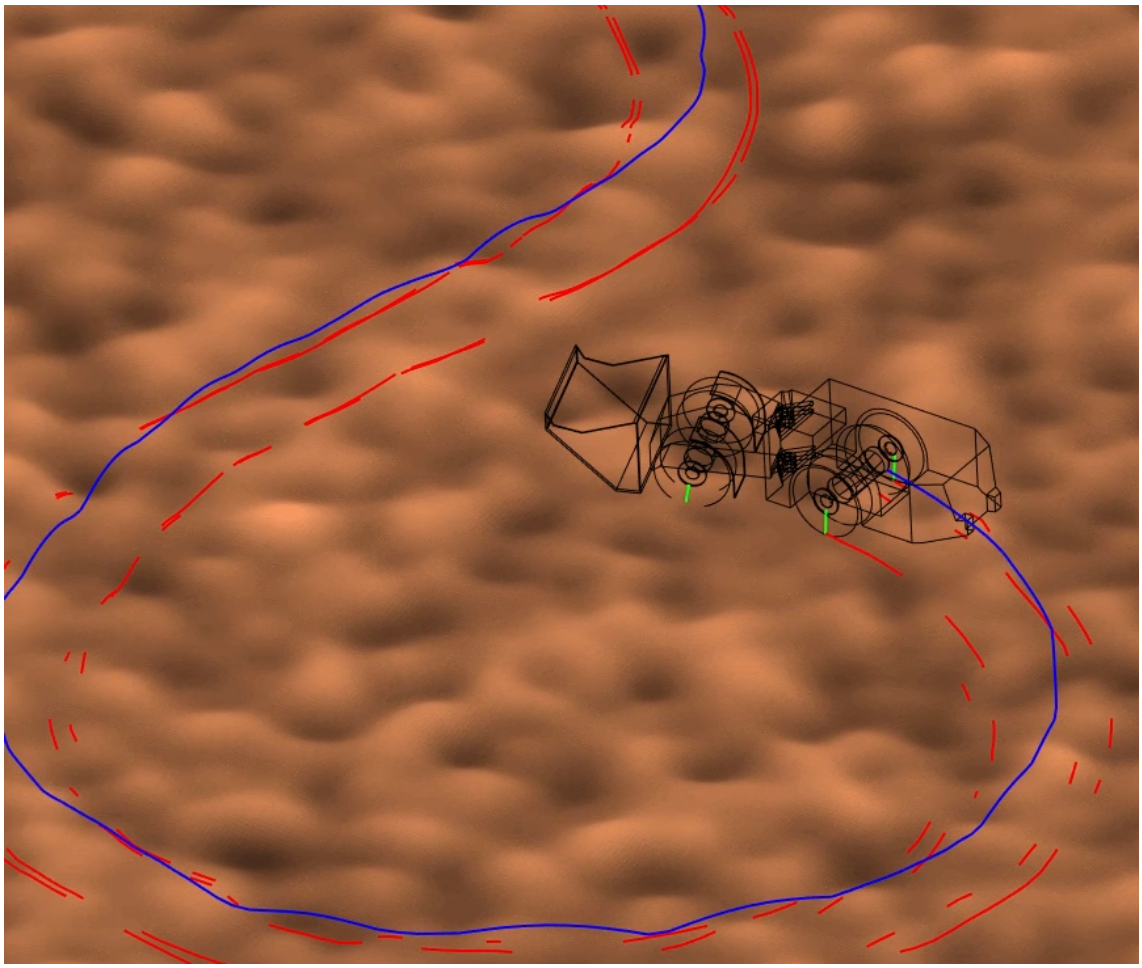
Algoritmin laskenta-ajat jäävät huomattavasti DPC-artikkelissa esitetyistä laskenta-ajoista, jotka olivat noin 50 metrin pituiselle matkalle alkuliikeradan laskennan osalta keskimäärin muutama sekunti ja myös muiden vaiheiden osalta alle 10 sekuntia. Tässä työssä käytettiin liikeratojen laskemiseen MATLAB-ohjelmistoa, jossa alkuliikeradan laskenta kesti noin 90 sekuntia. Heikompaan laskenta-aikaan on monia syitä, kuten esimerkiksi MATLAB, optimoimaton koodi, monimutkaisempi ajoneuvon asennon laskenta sekä liikeradan esitys ja liian tiheä pistepilvi. Liikeradan laskenta-aika ei välttämättä ole ongelma, koska tunneleissa reitit muuttuvat harvoin radikaalisti ja ne voidaan laskea rauhassa ennen tuotantovaihetta. Jos kuitenkin halutaan reaaliaikaisia väistöominaisuuksia, olisi liikeratojen laskentanopeuden oltava nopeampi.

4.3 Kaivoskoneen mallinnus ja simulointi WMRDE:llä

Mikäli kaarevuuspolynomit eivät ole riittävän tarkkoja liikeradan esittämiseen tai halutaan esimerkiksi simuloida reittiä nopeasti törmäyksien estämiseksi, voidaan käyttää WMRDE:tä. Tässä työssä kaivoskoneen mallinnuksessa on käytetty Seegmillerin väitöskirjassa [43] esitettyä menetelmää. Ensimmäiseksi osaksi on valittu takaosa, johon muut osat on kiinnitetty. Taka- ja etuosan välillä on ohjattava keskinivel. Etu-akseli on kiinteä toisin kuin taka-akseli. Hitausmomentit on pelkistetty sylintereiden ja suorakulmaisten särmiöiden avulla ja arvioitu datalehdeltä. Kuvissa 52 ja 53 on esimerkki simulaattorin toiminnasta.



Kuva 52. *WMRDE:llä voidaan mallintaa pyörällisiä ajoneuvoja. Kuvassa on kaivoskone mallinnettuna MATLAB-ympäristössä. Punaiset viivat kertovat pyörien aikaisemmat maakosketukset, sininen ajoneuvon sijainnin ja vihreät pyörien tämänhetkisen maakosketuksen.*



Kuva 53. *Suurilla nopeuksilla epätasaisessa maastossa maakosketus häviää. Ajoneuvo voi myös kaatua, jos maasto on liian kalteva, kaarrenopeus liian suuri tai painopiste liian korkealla.*

Kuvissa punaiset viivat ovat renkaiden maakosketukset, sininen ajoneuvon sijainti ja vihreät renkaiden hetkellinen maakosketus. Jälkimmäisestä kuvasta on havaittavissa, että maakosketus häviää, jos nopeus on suuri ja maasto on epätasaista. Simulaattorissa ei ole törmäyksiä tai kosketuksia muiden osien kanssa kuin pyörien, mutta ajoneuvo voi silti kaatua. Tämä voi tapahtua kaltevassa maastossa, liian suurella kaarrenopeudella tai liian korkealla painopisteellä.

Simulaattori käyttää yksinkertaista yhden pisteen maakosketuksen rengasmallia, joka ei vastaa todellisuutta etenkään epätasaisessa tai pehmeässä maastossa. Käytettäessä suoraan pistepilveä eikä 2,5D-maastoa, ajoneuvo hyppii helposti yhden pisteen maakosketuksen rengasmalleilla, koska pistepilvessä on paljon kohinaa. Tällöin pitäisi käyttää eri pyörämallia tai ainakin suodattaa pistepilveä enemmän. Mahdollisesti jatkotutkimuksissa tämän simulaattorin tuloksia pitää vielä verrata Mevean tekemän kaivoskonesimulaattorin tulosten kanssa, jos tätä simulaattoria halutaan hyödyntää esimerkiksi turvallisuuskerroksessa törmäysten estämiseksi. Mevea on suomalainen yritys, joka on yksi johtavia reaaliaikasimulaatioiden ja digitaalisten kaksosteknologioiden tuottajia. Tämä simulaattori ei tosin ainakaan vielä huomaa itse mahdollisia törmäyksiä ajoneuvon muiden osien kuin pyörien kanssa.

5. YHTEENVETO JA SUOSITUKSET

Tässä työssä esiteltiin, kuinka nykyistä kaivoskoneen navigointijärjestelmää voidaan parantaa. Nykyinen järjestelmä perustuu 2D-lasertutkaan, joka aiheuttaa monia ongelmia. Ensinnäkin ympäristömalli ja reitti joudutaan opettamaan, mikä vie aikaa. Reittiä ei voida laskea automaattisesti käyttäen vain 2D-ympäristömallia, koska ei ole olemassa tietoa, mitä mitatun seinän ja kuljetun reitin välillä on. 2D-lasertutkilla ei voida havaita mahdollisia esteitä mitatun tason ala- tai yläpuolelta. Tämän takia dynaamisissa ympäristöissä autonominen navigointi ei ole mahdollista. Lisäksi pahimmassa tapauksessa jokaiselle samassa ympäristössä liikkuvalle ajoneuvolle joudutaan opettamaan ympäristömalli ja reitti erikseen, jos lasertutkien korkeudet eivät ole samat.

3D-lasertutkat mahdollistavat yhden jatkuvasti päivitettävän ympäristömallin, joka on riittävän tarkka liikesuunnitteluun ainakin paikallisella tasolla. Lisäksi ajoneuvot voivat toimia myös dynaamisissa ympäristöissä. Ympäristömallin avulla reitit voidaan laskea myös automaattisesti. Ympäristömallina käytetään 3D-pistepilveä, johon ei tarvitse tehdä laskennallisesti raskasta pinnan rekonstruktiota. Ympäristömalli voidaan luoda pelkästään 3D-lasertutkiin ja ICP:hen perustuvalla SLAM-algoritmilla, jonka toimintavarmuutta voidaan parantaa muiden antureiden avulla. Ympäristömallin päivittäminen tapahtuu paikallisella tasolla yksittäisiin alikarttoihin, ja lisäksi kartasta voidaan poistaa dynaamiset kohteet.

Liikesuunnittelu perustuu käytännössä satunnaiseen näytteenottoon ja paikalliseen maaston arviointiin huomioiden ajoneuvon ominaisuudet sekä käyttäen ympäristömallin 3D-pistepilveä. Tässä työssä ei ole ratkaistu vielä, mikä olisi paras tapa paikalliseen reitin optimointiin kaivoskoneille.

Lisäksi tässä työssä esiteltiin nopea pyörällisille liikkuville roboteille tarkoitettu dynaaminen simulaattori, jota voidaan mahdollisesti myös käyttää jatkotutkimuksessa. Simulaattoria voidaan tarvita etenkin silloin, kun havaitaan, että käytettävien kaarevuuspolynomien tarkkuus ei ole riittävä. Simulaattoria voidaan mahdollisesti hyödyntää myös törmäyksien estämisessä.

3D-lasertutkien ja -pistepilvien käyttämisellä saavutetaan monia hyötyjä, joita ei aikaisemmalla järjestelmällä ole mahdollista saavuttaa. Näitä hyötyjä ovat seuraavat:

1. Yksi jatkuvasti päivittyvä ympäristömalli sopii jokaiselle ajoneuvolle.
2. Reittejä voidaan suunnitella automaattisesti ja niitä voidaan myös muokata tienpinnan kulumisen estämiseksi.

3. Ajoneuvoja voi mahdollisesti käyttää dynaamisissa ympäristöissä.

5.1 Jatkotutkimus- ja kehitystarpeet

Tässä työssä käytetyt algoritmit ja näiden muokkaukset tulee täydentää tarvittavilta osilta. Täydennykset koskevat etenkin SLAM-algoritmia ja paikallista optimointia. SLAM-algoritmin tulisi ottaa huomioon mahdollisesti lasertutkan liikkeestä johtuvien virheiden korjaaminen, dynaamisten pisteiden luokittelu sekä poistaminen ja alikarttojen päivittäminen. Liikeradan paikallinen optimointi ja mahdollisesti jonkinlainen tasoitusalgoritmi vaativat jatkotutkimusta. Niiden lisäksi olisi hyvä miettiä, kuinka tunneleiden seinät ja etäisyydet näihin otettaisiin huomioon liikesuunnittelun eri vaiheissa. Kuitenkaan ympäristöt eivät aina rajoitu pelkästään tunneleihin, joten olisi hyvä myös miettiä, kuinka erilaiset ympäristöt, kuten tunnelit, avoimet maastot ja tiet otetaan liikesuunnittelussa huomioon. Jos esimerkiksi halutaan rajata ajoneuvon liikerata pelkästään tielle ja etenkin sen tietylle kaistalle eikä sitä ole rajattu mitenkään avoimesta maastosta, algoritmi tuottaa luultavasti ei-toivottuja tuloksia palauttamalla liikeratoja, jotka oikaisevat maaston antamissa puitteissa. Liikesuunnittelussa olisi hyvä myös hyödyntää mahdollisimman paljon aikaisemmin laskettuja liikeratoja.

Kun algoritmit on saatu riittävälle tasolle, ne tulee testata joko Mevean tekemällä kaivoskonesimulaattorilla tai vastaavalla. Samalla voidaan kokeilla erilaisia lasertutkien kokoonpanoja kaivoskoneissa. Pyrkimyksenä olisi saada kaivoskoneen ympäristöstä riittävän peittävä alue, jolla algoritmit toimivat. Jatkotutkimusta tarvitaan myös, kuinka vanhan kalusto ja uusi ympäristömalli saadaan toimimaan yhdessä mahdollisimman pienillä kustannuksilla. Paikannus voisi tapahtua partikkelisuodattimen avulla hyödyntämällä 2D-lasertutkien mittausdataa ja esimerkiksi erillisellä ajoneuvolla kerättyä 3D-pistepilveä. Jokaisessa ajoneuvossa ei tällöin tarvitsisi olla kalliimpia 3D-lasertutkia, vaan pistepilvi tuotettaisiin yksittäisellä ajoneuvolla ja päivitettäisiin tarvittaessa. Tällöin kuitenkin 2D-lasertutkilla varustetut ajoneuvot ovat suuremmassa vaarassa törmätä etenkin dynaamisiin kohteisiin eikä karttojen päivitys onnistu.

Pistepilviä voi hyödyntää muutenkin jatkotutkimuksessa, kuten tekoälyssä tai anturifuusiossa. Navigointijärjestelmä voisi perustua kokonaisuudessaan tekoälyyn. Simulaattorin avulla voitaisiin opettaa navigointijärjestelmää toimimaan erilaisissa ympäristöissä. Tekoälyyn perustuvien järjestelmien toiminnan ymmärtäminen voi olla kuitenkin haastavaa, mikä tekee mahdollisten virheiden etsinnästä ja poistamisesta vaikeaa.

Anturifuusiossa yhdistetään useista eri lähteistä saatua mittausdataa tai laskettua dataa toisiinsa, jolloin saavutetaan pienempi epävarmuus mitattavasta kohteesta. Tässä työssä on käytetty ICP:hen perustuvaa SLAM-algoritmia. ICP ei toimi kovinkaan hyvin, jos mitattava ympäristö ei sisällä helposti havaittavia kohteita. ICP ei kykene tuottamaan luotettavia tuloksia esimerkiksi suorakulmaisessa pitkässä käytävässä, jolloin sen avulla

muodostettuun pistepilveen syntyy virhettä. Esimerkiksi jo pelkästään matkamittarin lisäämisellä voidaan saavuttaa huomattavia parannuksia.

Pistepilvidataa voidaan hyödyntää myös muihin tarkoituksiin kuin pelkästään navigointiin. Esimerkiksi sen avulla voitaisiin selvittää keskinivelen kulman suuruus käyttämällä ajoneuvon 3D-mallia ja vertaamalla, mikä asento tuottaa parhaiten pistepilvessä tietyllä alueella olevat pisteet. Käyttämällä aikaisempia asentoja hyväksi, voidaan mahdollisesti nopeuttaa asennon etsimistä. Näin saadaan poistettua myös navigoinnin kannalta turhat pisteet yksittäisistä pistepilvistä. Pistepilvien ja kameradatan avulla voidaan myös mahdollisesti tunnistaa ja seurata paremmin muita kohteita ympäristössä, kuten ihmisiä. Tällöin ajoneuvo, operaattori tai toiminnan valvoja on tietoisempi ajoneuvon ympäristöstä.

5.2 Työn onnistuminen

Käyttämällä tässä työssä esitettyjä menetelmiä on mahdollista saavuttaa ensimmäisessä luvussa mainitut tavoitteet kaivoskoneen autonomisessa navigoinnissa. Merkittävimmät parannukset nykyiseen järjestelmään verrattuna ovat seuraavat:

- Ympäristöt pystytään pitämään automaattisesti ajan tasalla.
- Järjestelmä kykenee määrittämään ajoneuvon reitit ilman merkittäviä toimia operaattorilta.
- Järjestelmää voidaan käyttää dynaamisissa ympäristöissä.

Edellisessä alaluvussa on käsitelty lisätutkimuskohteet. Omasta mielestäni työni on kuitenkin erittäin onnistunut, ja olen erittäin tyytyväinen lopputulokseen. Työni ei näytä lainkaan niin yksinkertaiselta kuin aluksi kuvittelin sen olevan. Lisäksi ajattelin sen koostuvan pääasiassa 2D-reitinsuunnittelualgoritmeista. Siten aikaakin on kulunut paljon enemmän kuin aluksi ajattelin. Työni on antanut minulle kattavan kuvan itsenäiseen navigointiin liittyvistä ongelmista, teknologioista ja käytössä olevista menetelmistä. Työn tekemisessä kertyneiden tietojen avulla voin tarvittaessa jatkaa luottavaisin mielin aiheen parissa.

LÄHTEET

[1] C. Altafini, Why to use an articulated vehicle in underground mining operations? IEEE International Conference on Robotics and Automation, May 10–15, 1999, IEEE, Detroit, Michigan, USA, pp. 3025 vol.4.

[2] K. Amdahl, M. Lundström, Automatic truck saves money underground, World Mining, Vol. 25, No. 12, 1972, pp. 40–44.

[3] Automine® Mine Equipment Automation & Teleoperation Systems — Sandvik Mining and Rock Technology, Sandvik, web page. Available (accessed 22.3.2018): https://www.rocktechnology.sandvik/en/products/automation/automine-equipment-and-teleoperation-systems/.

[4] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, 1992, pp. 239–256.

[5] R.H. Byrd, J.C. Gilbert, J. Nocedal, A trust region method based on interior point techniques for nonlinear programming, Mathematical Programming, Vol. 89, No. 1, 2000, pp. 149–185.

[6] R.H. Byrd, M.E. Hribar, J. Nocedal, An Interior Point Algorithm for Large-Scale Nonlinear Programming, SIAM Journal on Optimization, Vol. 9, No. 4, 1999, pp. 877–900.

[7] Cameco 2015 Online Annual Report, Cameco, web page. Available (accessed 3.11.2017): https://www.cameco.com/annual_report/2015/images/mda/our-operations-and-projects/cigarlake-mine.jpg.

[8] N.W. Campbell, B.T. Thomas, Lane Boundary Tracking for an Autonomous Road Vehicle, in: D. Hogg, R. Boyle (ed.), BMVC92, Springer-Verlag London Limited, London, 1992, pp. 157–166.

[9] Y. Chen, G. Medioni, Object modelling by registration of multiple range images, Image and Vision Computing, Vol. 10, No. 3, 1992, pp. 145–155.

[10] D. Collins, How encoder resolution is determined, WTWH Media, web page. Available (accessed 20.5.2018): https://www.linearmotiontips.com/how-encoder-resolution-is-determined/.

[11] R.P.G. Collinson, Introduction to Avionics Systems, 3rd ed., Springer Science + Business Media, Springer Dordrecht Heidelberg London New York, 2011, pp. 530.

[12] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments, The International Journal of Robotics Research, Vol. 29, No. 5, 2010, pp. 485–501.

[13] B.J. Dragt, Modelling and control of an autonomous underground mine vehicle, Master's, University of Pretoria, 2007, pp. 118. Available: http://upetd.up.ac.za/thesis/available/etd-08282007-101213.

[14] P. Duddu, Ten technologies with the power to transform mining, Mining Technology, web page. Available (accessed 3.11.2017): http://www.mining-technology.com/features/featureten-technologies-with-the-power-to-transform-mining-4211240/.

[15] G. Eriksson, A. Kitok, Automatic loading and dumping using vehicle guidance in a Swedish mine, International Symposium on Mine Mechanisation and Automation, June 10–13, 1991, Colorado School of Mines, Golden, Colorado, USA, pp. 15.33–15.40.

[16] K. Haroon, Ring Laser Gyro - Principle of Operation, Airline Pilots Forum, web page. Available (accessed 27.5.2018): https://www.theairlinepilots.com/forum/viewtopic.php?t=909.

[17] P.E. Hart, N.J. Nilsson, B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, Vol. 4, No. 2, 1968, pp. 100–107.

[18] R. Hewitt, Applying FastSLAM to articulated rovers, Master's, Carleton University, 2012, pp. 197. Available: https://curve.carleton.ca/c251d329-4546-4859-8ca0-710522302a89.

[19] R. Hurteau, M. St-Amant, Y. Laperriere, G. Chevrette, Optical guidance system for underground mine vehicles, IEEE International Conference on Robotics and Automation, May 12–14, 1992, IEEE, Nice, France, pp. 644 vol.1.

[20] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, The International Journal of Robotics Research, Vol. 30, No. 7, 2011, pp. 846–894.

[21] P. Krüsi, P. Furgale, M. Bosse, R. Siegwart, Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments, Journal of Field Robotics, Vol. 34, No. 5, 2017, pp. 940–984.

[22] S.M. LaValle, Rapidly-Exploring Random Trees: A New Tool for Path Planning, Report No. TR 98-11, Computer Science Department, Iowa State University, 1998, pp. 4.

[23] S.M. LaValle, J.J. Kuffner, Randomized Kinodynamic Planning, The International Journal of Robotics Research, Vol. 20, No. 5, 2001, pp. 378–400.

[24] H.C. Lefèvre, The Fiber-Optic Gyroscope, 2nd ed., Artech House, Boston|London, 2014, pp. 416.

[25] J.J. Leonard, H.F. Durrant-Whyte, Simultaneous map building and localization for an autonomous mobile robot, IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91, November 3–5, 1991, IEEE, Osaka, Japan, pp. 1442–1447 vol.3.

[26] LKAB - Our underground mines, LKAB, web page. Available (accessed 3.11.2017): https://www.lkab.com/en/about-lkab/from-mine-to-port/mining/our-underground-mines/.

[27] H. Mäkelä, Overview of LHD navigation without artificial beacons, Robotics and Autonomous Systems, Vol. 36, No. 1, 2001, pp. 21–35.

[28] H. Mäkelä, H. Lehtinen, K. Rintanen, K. Koskinen, Navigation system for LHD machines, 2nd IFAC Conference on Intelligent Autonomous Vehicles (IAV'95), June 12–14, 1995, Finnish Society of Automation, Espoo, Finland, pp. 314–319.

[29] D. McFadden, Schematic of a fibre-optic-gyroscope based on the sagnac effect, Wikimedia Commons, web page. Available (accessed 20.5.2018): https://commons.wikimedia.org/wiki/File:Fibre-optic-interferometer.svg.

[30] M. Montemerlo, S. Thrun, FastSLAM – A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics, 1st ed., Springer-Verlag Berlin Heidelberg, New York, 2007, pp. 120.

[31] B. Nagy, A. Kelly, Trajectory Generation for Car-like Robots Using Cubic Curvature Polynomials, 3rd International Conference on Field and Service Robotics, June 11–13, 2001, Helsinki, Finland, pp. 6.

[32] A. Nash, K. Daniel, S. Koenig, A. Felner, Theta*: Any-Angle Path Planning on Grids, Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), July 22–26, 2007, Vancouver, British Columbia, The AAAI Press, Menlo Park, California, USA, pp. 1177–1183.

[33] T.M. Nayl, On Autonomous Articulated Vehicles, Dissertation, Luleå University of Technology, 2015, pp. 166. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2:990305.

[34] E. Nelson, Berkeley Localization and Mapping, GitHub, web page. Available (accessed 30.5.2017): https://github.com/erik-nelson/blam.

[35] J. Nocedal, S. Wright, Numerical Optimization, 2nd ed., Springer-Verlag New York, USA, 2006, pp. 664.

[36] M. Ovsjanikov, PS 1 - Rigid shape registration, Computer Science Laboratory of the École polytechnique, web page. Available (accessed 31.3.2018): http://www.lix.polytechnique.fr/~maks/Verona_MPAM/TD/TD1/.

[37] F. Pomerleau, F. Colas, R. Siegwart, S. Magnenat, Comparing ICP variants on real-world data sets, Autonomous Robots, Vol. 34, No. 3, 2013, pp. 133–148.

[38] G. Ramalingam, T. Reps, An Incremental Algorithm for a Generalization of the Shortest-Path Problem, Journal of Algorithms, Vol. 21, No. 2, 1996, pp. 267–305.

[39] Sandvik LH410 — Technical Specification, Sandvik, web page. Available (accessed 11.4.2017): https://dokumen.site/file/ts410-a5b39ef69bb254.

[40] Sandvik LH410 Underground Loader – Technical Specification, Sandvik, web page. Available (accessed 11.4.2017): http://unitedminingrentals.com/pdf/trucks/LH410.pdf.

[41] Sandvik TH663 — Technical Specification, Sandvik, web page. Available (accessed 29.8.2017): http://unitedminingrentals.com/pdf/trucks/TH663.pdf.

[42] S. Scheding, G. Dissanayake, E.M. Nebot, H. Durrant-Whyte, An experiment in autonomous navigation of an underground mining vehicle, IEEE Transactions on Robotics and Automation, Vol. 15, No. 1, 1999, pp. 85–95.

[43] N.A. Seegmiller, Dynamic Model Formulation and Calibration for Wheeled Mobile Robots, Dissertation, Carnegie Mellon University, 2014, pp. 126. Available: http://repository.cmu.edu/dissertations/460.

[44] A.V. Segal, D. Haehnel, S. Thrun, Generalized-ICP, Proceedings of Robotics: Science and Systems V, Seattle, USA, June 28 – July 1, 2009, MIT Press, Cambridge, Massachusetts, pp. 8.

[45] R.C. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, The International Journal of Robotics Research, Vol. 5, No. 4, 1986, pp. 56–68.

[46] H. Sönnerlind, Modeling the Dynamics of a Gyroscope, COMSOL, web page. Available (accessed 24.5.2018): https://www.comsol.com/blogs/modeling-the-dynamics-of-a-gyroscope/.

[47] O. Sorkine-Hornung, M. Rabinovich, Least-Squares Rigid Motion Using SVD, ETH Zurich, Department of Computer Science, Institute of Visual Computing, web page. Available (accessed 5.3.2018): https://igl.ethz.ch/projects/ARAP/svd_rot.pdf.

[48] A. Stentz, Optimal and efficient path planning for partially-known environments, Proceedings of the 1994 IEEE International Conference on Robotics and Automation, May 8–13, 1994, IEEE, San Diego, CA, USA, pp. 3310–3317 vol.4.

[49] STMicroelectronics - L3G3250A 3-Axis MEMS Gyroscope, SYSTEM PLUS CONSULTING, web page. Available (accessed 20.5.2018): http://www.systemplus.fr/reverse-costing-reports/st-l3g3250a-3-axis-mems-gyroscope/.

[50] D.H. Titterton, J.L. Weston, Institution of Electrical Engineers, Strapdown inertial navigation technology, 2nd ed., Institution of Electrical Engineers, Stevenage, 2004, pp. 558.

[51] VLS-128, Velodyne LiDAR, web page. Available (accessed 29.6.2018): https://velodynelidar.com/vls-128.html.

[52] R.A. Waltz, J.L. Morales, J. Nocedal, D. Orban, An interior algorithm for nonlinear optimization that combines line search and trust region steps, Mathematical Programming, Vol. 107, No. 3, 2006, pp. 391–408.

[53] N. Zikos, V. Petridis, 6-DoF Low Dimensionality SLAM (L-SLAM), Journal of Intelligent & Robotic Systems, Vol. 79, No. 1, 2015, pp. 55–72.